VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

SYNTÉZA PRAVDĚPODOBNOSTNÍCH PROGRAMŮ S OPTIMÁLNÍ CENOU

SYNTHESIS OF PROBABILISTIC PROGRAMS WITH REWARDS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **VOJTĚCH HRANIČKA**
AUTHOR

VEDOUCÍ PRÁCE — **RNDr. MILAN ČEŠKA, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Hranička Vojtěch**
Program: Informační technologie
Název: **Syntéza pravděpodobnostních programů s optimální cenou**
Synthesis of Probabilistic Programs with Rewards
Kategorie: Formální verifikace

Zadání:

1. Nastudujte existující metody pro automatizovaný návrh a syntézu pravděpodobnostních programů s důrazem na metody založené na MDP abstrakci a protipříkladech.
2. Vyhodnoťte tyto metody na relevantních příkladech a identifikujte nedostatky těchto metod v kontextu syntézy pravděpodobnostních programů s optimální cenou.
3. Navrhněte vylepšení a rozšíření těchto metod dovolující efektivní syntézu programů s optimální cenou. Navrhněte podporu specifikace používající tzv. rewards.
4. Implementujte navržené vylepšení a rozšíření v rámci existujících nástrojů pro analýzu pravděpodobnostních programů (např. STORM nebo PRISM).
5. Proveďte podrobné experimentální vyhodnocení navržených metod na vhodné sadě problémů.

Literatura:

1. Milan Češka, Nils Jansen, Sebastian Junges, and Joost-Pieter Katoen. Shepherding hordes of Markov chains. In Proc. of TACAS'19. Springer, 2019.
2. Milan Češka, Christian Hensel, Sebastian Junges, and Joost-Pieter Katoen. Counterexample-Driven Synthesis for Probabilistic Program Sketches. In Proc. of FM'19. Springer, 2019.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a částečně 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Češka Milan, RNDr., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 11. listopadu 2020

Abstrakt

Tato práce se zabývá syntézou pravděpodobnostních modelů s optimální cenou. Pravděpodobnostní syntéza slouží k automatickému návrhu systému, který splňuje požadované specifikace. V této práci se věnuji způsobu syntézy kde máme šablonu pro daný systém, která obsahuje neznámé části a cílem je najít takovou kombinaci nastavení daných částí tak, aby výsledný systém splňoval specifikované požadavky. V poslední době se objevují nové přístupy uvažující o množině řešení jako o rodině Markovových řetězců. Jedním z těchto přístupů je použití nové metody kombinující metody protipříklady řízeného zjemňování abstrakce a induktivní syntézy. Tato metoda svou efektivitou převyšuje ostatní metody pro pravděpodobnostní syntézu. V této práci se konkrétně zaměřuji na rozšíření specifikačního jazyka tohoto nástroje o možnost použití takzvaných rewardů a until vlastností. Díky těmto rozšířením je možné lépe a jednodušeji specifikovat hledané řešení. Experimenty demonstrují, že i po rozšíření daného nástroje o tyto možnosti specifikace jeho rychlost v porovnání se standardní metodou syntézy zůstává až o několik řádů efektivnější.

Abstract

This thesis pursues the synthesis of probabilistic programs with rewards. Probabilistic synthesis leads to the automatic proposal of a system which fulfills required specifications. In this thesis, I work with a form of synthesis where we have a sketch of a given system. This sketch includes unknown variables and the objective is to find a combination of configuration of given variables in order to have the final program meeting the specified requirements. Recently, new approaches considering a set of solutions as a family of Markov chain have appeared. One of these approaches is the usage of a new method combining counterexample-guided abstraction refinement and counterexample-guided inductive synthesis. This method exceeds other methods for synthesis of probabilistic programs with its efficiency. In this thesis, I concretely focus on extending this tool's specification language by adding a possibility of application of rewards and until properties. Thanks to these extensions it is possible to specify searched solution more efficiently. Experiments demonstrate that even after the addition of these possibilities of specifications the speed of a given tool remains by a margin of order of magnitudes more effective than the standard method of synthesis.

Klíčová slova

Syntéza pravděpodobnostních modelů, Markovovy modely

Keywords

Synthesis of probabilistic models, Markov models

Citace

HRANIČKA, Vojtěch. *Syntéza pravděpodobnostních programů s optimální cenou.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce RNDr. Milan Češka, Ph.D.

Syntéza pravděpodobnostních programů s optimální cenou

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana RNDr. Milana Češky, Ph.D. Další informace mi poskytl Ing. Roman Andriushchenko. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Vojtěch Hranička
17. května 2021

Poděkování

Chtěl bych poděkovat vedoucímu práce Milanovi Češkovi za odborné rady a skvělé vedení práce. Dále bych chtěl poděkovat Romanovi Andriushchenkovi za ochotné zodpovídání mých dotazů ohledně nástroje PAYNT.

Obsah

Kapitola 1

Úvod

Při vývoji technologií se v inženýrství často využívají pravděpodobnostní modely. Pomocí těchto modelů je možné analyzovat systémy, které vykazují náhodné chování. Jejich využití lze najít v mnoha odvětvích, jako jsou například biologie [12], návrh komunikačních protokolů [9] nebo verifikace randomizovaných algoritmů [11]. K analýze těchto modelů existuje mnoho nástrojů, například nástroje PRISM [9] nebo Storm [5]. Při vývoji systému často nastává situace, kdy máme pouze částečné řešení problému a současně komponenty, které je možné použít k výslednému řešení. Kombinováním různých komponentů vzniká množina řešení. Z těchto řešení chceme vybrat takové, které splňuje požadované specifikace. Tímto problémem se zabývá syntéza pravděpodobnostních programů. V poslední době se objevují nové přístupy, uvažující o množině řešení jako o rodině Markovových řetězců. Pokud pro popis pravděpodobnostního procesu použijeme Markovovy řetězce, tak úkolem syntézy je analyzovat všechny řetězce zadané rodiny Markovových řetězců a následně zjistit, který řetězec splňuje požadované specifikace, popřípadě zjistit, že žádný z řetězců požadované specifikace nesplňuje.

Tato práce se zaměřuje na novou hybridní metodu kombinující induktivní a deduktivní přístup [3][1]. Díky tomuto přístupu svou efektivitou převyšuje ostatní metody pro pravděpodobnostní syntézu. Časová složitost řešení složitých problémů se díky tomuto přístupu zmenší oproti metodě one-by-one [4] z řádu dní na řády minut. Na vývoji nástroje pro pravděpodobnostní syntézu pomocí této hybridní metody pracuje tým na FIT VUT ve spolupráci s Německem a USA.

Cílem této práce je rozšíření specifikačního jazyka výše zmíněného nástroje, tak aby umožňoval použití v kontextu syntézy pravděpodobnostních programů s optimální cenou, konkrétně možnost použití takzvaných rewardů. To umožňuje přiřazovat jednotlivým stavům hodnotu ceny a používat specifikace týkající těchto cen. Například limit průměrné ceny stavů cesty do cílového stavu. Druhou oblastí této práce je specifikační jazyk rozšířit o možnost specifikace podmínek pomocí until vlastností. Díky tomuto rozšíření je možné specifikovat vlastnosti podmíněné platností jiných podmínek a tak vytvářet konkrétnější specifikace. Navrženou metodu demonstruji mimo jiné na modelu Hermanova algoritmu [10]. Při pravděpodobnostní syntéze tohoto modelu je potřeba prohledat přes 3 milióny možných řešení, která jsou tvořena průměrně 1153 stavy. Při použití této metody pravděpodobnostní syntéza tohoto modelu trvala 141 sekund a nalezení optimálního řešení trvalo 36 minut.

[1] přijato do TACAS 2021

První část práce se zabývá teorií potřebnou k pochopení hlavní části práce. Dále je popsán návrh a implementace rozšíření metody pravděpodobnostní syntézy o možnost použití rewardů a until vlastností. Poté následuje popis experimentálního vyhodnocení implementovaných rozšíření.

Kapitola 2

Teorie

Tato kapitola se zabývá shrnutím důležitých znalostí potřebných pro pochopení navazujících kapitol. Nejdříve se zde budu zabývat popisem diskrétních Markovových řetězců a Markovových reward modelů. Následně se věnuji způsobům zjišťování, zda daný diskrétní Markovův řetězec nebo reward model odpovídá zadané specifikaci pomocí model checkingu. Dále následuje stručný popis Rodin Markovových řetězců. Poté se zabývám protipříklady a Markovovými rozhodovacími procesy. V závěru této kapitoly se věnuji problematice syntézy pravděpodobnostních programů a vysvětlení již existujících metod pro její provedení.

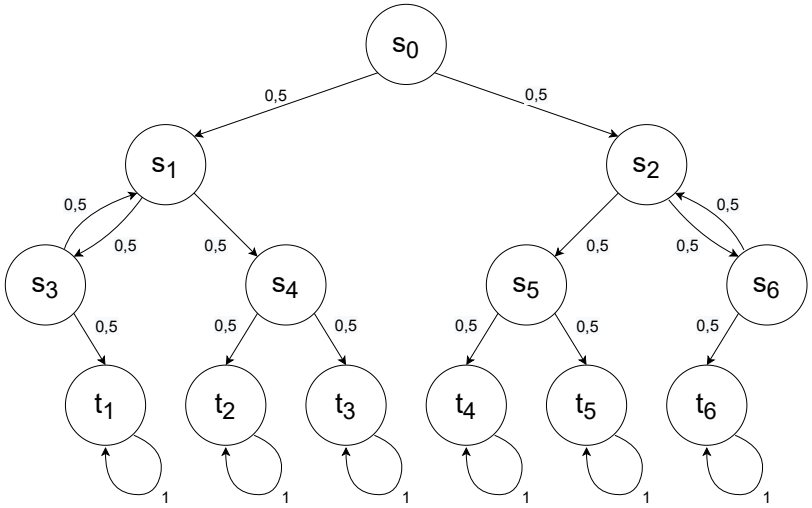
2.1 Diskrétní Markovovy řetězce (DTMC)

Definice 2.1.1 ***(DTMC).*** *Diskrétní Markovův řetězec D [14] je trojice (S, s_{init}, P), kde S představuje množinu všech stavů daného řetězce, s_{init} představuje počáteční stav řetězce, pro který platí $s_{init} \in S$ a $P = S \times S \rightarrow [0, 1]$ je množina pravidel pro přechod mezi stavy. Zápis $P(s, s')$ značí pravděpodobnost přechodu ze stavu s do stavu s'. Pro každé $s \in S$ platí $\sum_{s' \in S} P(s, s') = 1$. Množinu všech rozdělení pravděpodobnosti nad množinou S budeme dále značit $Distr(S)$.*

Cesta DTMC je posloupnost stavů DTMC pro které platí $P(s, s') > 0$. Pravděpodobnost cesty π se rovná součinu pravděpodobností všech přechodů mezi stavy, které jsou součástí dané cesty. Tedy $P(\pi) = \prod_{i>0} P(s_{i-1}, s_i)$.

Příklad 2.1.2 *Představme si situaci, kdy chceme modelovat simulaci hodu šestistěnnou kostkou pomocí série hodů mincí. Pro tento účel vytvoříme DTMC $\mathcal{D} = (S, s_0, P)$ který je graficky znázorněn na obrázku 2.1. Množina stavů S v tomto případě obsahuje celkem 13 stavů. Konkrétně 7 stavů začínajících písmenem* `s` *slouží k simulování hodů mincí. Z těchto stavů tedy vedou dva přechody s pravděpodobností $0,5$, symbolizující rozhodnutí na základě padnutí panny nebo orla. Počátečním stavem je v tomto případě stav s_0, který značí situaci, kdy nebyl proveden žádný hod mincí. Zbývajících 6 stavů začínajících písmenem* `t` *značí výsledek simulace. Tyto stavy jsou absorbující a simulace v nich končí. Například stav t_1 značí výsledné číslo 1.*

Jednou z možných cest pro tento model by byla například cesta π kde $\pi = s_0 s_1 s_3 t_1$. Pravděpodobnost takové situace se dá vypočítat vynásobením pravděpodobností všech přechodů této cesty. V tomto případě by to tedy bylo $|\pi| = P(s_0, s_1) \cdot P(s_1, s_3) \cdot P(s_3, t_1)$, což se rovná $0,5 \cdot 0,5 \cdot 0,5 = 0,125$. Pravděpodobnost cesty π je tedy v tomto případě $0,125$. Pokud by platilo $\pi = s_0, s_1, s_2$, tak π nemůže být cestou, protože $P(s_1, s_2) = 0$ a tedy nesplňuje podmínku $P(s, s') > 0$, která musí platit pro všechny přechody mezi stavy dané cesty.



Obrázek 2.1: DTMC modelující hod kostkou simulovaný pomocí série hodů mincí

Pro DTMC platí Markovovo pravidlo. Markovovo pravidlo znamená, že chování v každém stavu není ovlivněno předchozími stavy ve kterých se model nacházel, ale pouze jeho aktuálním stavem. To znamená, že rozdělení pravděpodobností přechodů je pro konkrétní stav stejné pro všechny cesty π.

Pokud máme DTMC $D = (S, s_{init}, P)$, tak následovníkem stavů $C \in S$ nazýváme všechny stavy, do kterých je pravděpodobnost přechodu ze stavu C nenulová. Tyto stavy značíme $Succ(C)$. Platí že $Succ(C) := \{t \in S \mid \exists s \in C.P(s,t) > 0\}$.

Definice 2.1.3 ***(Podřetězec DTMC [14]).*** *Mějme DTMC $D = (S, s_{init}, P)$ a kritické stavy C, pro které platí $C \subseteq S$ a zároveň $s_{init} \in C$. Podřetězec $D \downarrow C = (S', s_{init}, P')$, kde $S' = C \cup Succ(C)$ a pro P' platí $P'(s,t) = P(s,t)$ pokud $s \in C$, $P'(s,s) = 1$ pokud $s \in Succ(C)$ a v ostatních případech $P'(s,t) = 0$.*

2.1.1 Markovův rewards model (MRM)

Při práci s DTMC je někdy potřeba přiřadit stavům určitou cenu (rewardy) [13]. Rewardy podle použití mohou symbolizovat například spotřebu baterie, finanční cenu, časový zpoždění nebo paměťovou náročnost. Toto rozšíření DTMC umožňuje tvorbu výrazně jednodušších modelů, protože bez použití rewardů bychom pro modelování cen museli pro každý stav vytvořit velké množství stavů pro každou možnou hodnotu měřené ceny.

Definice 2.1.4 ***(MRM).*** *Markovův reward model (MRM) je dvojice $\mathcal{M} = (D, rew)$, kde D je DTMC $D = (S, s_0, P)$, tak jak je popsáno v 2.1.1 a rew je funkce přiřazující stavům jejich rewardy $rew : S \to \mathbb{R}_{\geqslant 0}$.*

Celková hodnota rewardů cesty π se rovná součtu rewardů všech stavů, které jsou součástí dané cesty. Tedy $rew(\pi) = \sum_{i>0}^{n-1} rew(s_i)$.

Příklad 2.1.5 *Mějme DTMC D z příkladu 2.1.2. Představme si MRM $\mathcal{M} = (D, rew)$, kde pro všechny stavy s_x platí $rew(s_x) = 1$ a pro všechny stavy t_x platí $rew(s_x) = 0$. Výpočet hodnoty rewardu cesty π kde $\pi = s_0 s_1 s_3 s_1 s_4 t_3$ se rovná součtu hodnot rewardů všech stavů cesty π. Tedy v tomto případě je pravděpodobnost cesty π rovna $|\pi| = 1+1+1+1+1+0 = 5$.*

2.2 Model checking

Model checking [8] je kontrola, zda DTMC splňuje požadovanou vlastnost. V této práci se budeme zabývat pravděpodobnostními vlastnostmi a reward vlastnostmi. Jedním ze způsobů popsání pravděpodobnostních vlastností je použití PCTL logiky. Tento způsob zápisu pravděpodobnostních vlastností je použit i v této práci a je detailněji popsán v následující části.

2.2.1 PCTL logika

Definice 2.2.1 *Syntaxe PCTL následuje tyto pravidla:*

$$\Phi ::= true \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{P}_{\bowtie\lambda}[\phi]$$
$$\phi ::= \mathbf{X}\Phi \mid \Phi\mathbf{U}^{\leq k}\Phi$$

Znak $\bowtie$ zde představuje jednu z možností: $<, \leq, >, \geq$, znak λ značí hraniční pravděpodobnost této vlastnosti, kdy $\lambda \in [0,1]$ a $k \in \mathbb{N} \cup \{\infty\}$.

Ve výše uvedené definici je důležité rozlišovat stavové formule Φ a formule cest ϕ. Zatímco stavové formule Φ se používají ke specifikaci vlastností DTMC, formule ϕ slouží pouze jako parametr pro operátor $\mathcal{P}_{\bowtie\lambda}[\cdot]$. Diskrétní Markovův řetězec splňuje $\mathcal{P}_{\bowtie\lambda}[\phi]$ právě tehdy, pokud pravděpodobnost splnění formule ϕ odpovídá požadavkům dané vlastnosti $\bowtie \lambda$. Ta standardně obsahuje jeden ze dvou různých operátorů. Prvním z nich je operátor $\mathbf{X}\Phi$, který značí že je formule Φ platná v příštím stavu, tedy po provedení jednoho kroku. Druhou možností je operátor $\Phi\mathbf{U}^{\leq k}\Psi$. Ten platí v případě, kdy je možné do k kroků splnit podmínku Ψ, přičemž je v každém stavu splněna podmínka Φ. Tento operátor budeme dále nazývat `bounded until`. Variantu toho to operátoru pro kterou platí $k = \infty$ nazýváme `unbounded until`. Pro zjednodušení zápisu se u formulí cest používá také operátor $\Diamond$. Ten v kontextu $\Diamond\Phi$ znamená, že podmínku Φ je možné splnit. Zápis $\Diamond^{\leq k}\Phi$ značí, že podmínku Φ je možné splnit do počtu k kroků. Případ kdy stav s splňuje PCTL formuli Φ se značí $s \models \Phi$. V opačném případě zapisujeme $s \not\models \Phi$. Obdobně je to u cesty π a PCTL formule ϕ, kde píšeme $\pi \models \phi$.

Vlastnosti $\varphi = P_{\bowtie\lambda}[\Diamond\, T]$ pro které platí $\bowtie \in \{\leq, <\}$, budeme dále označovat jako safety vlastnosti. Vlastnosti pro které platí naopak $\bowtie \in \{\geq, >\}$,budeme dále označovat jako liveness vlastnosti.

2.2.2 Model checking pro DTMC

Pro zjištění pravděpodobnosti dosažení množiny cílových stavů T z libovolného stavu s je potřeba zjistit pravděpodobnost všech možných cest mezi stavy s a T a tyto pravděpodobnosti sečíst. To není problém při omezené délce použitých cest. Pokud by ale délka použitých cest byla neomezená, zapříčinilo by to, že cest bude neomezené množství a tento způsob nebude možné použít. V takovém případě se pro zjištění pravděpodobnosti dosažení daného stavu používá algoritmus 1 [9], kde se výsledek získá vypočtením soustavy rovnic.

Příklad 2.2.2 *Uvažujeme DTMC z příkladu 2.1.2. Chceme zjistit pravděpodobnost dosažení stavu t_1 z počátečního stavu s_0. Pro výpočet potřebujeme vytvořit soustavu rovnic za pomocí algoritmu 1. Vyřešením této soustavy rovnic, docházíme k výsledku $P[s_0 \models \Diamond\{t_3\}] = \frac{1}{6}$.*

Algoritmus 1: Algoritmus pro získání pravděpodobnosti dosažení cílového stavu DTMC

Input: DTMC $D = (S, s_0, P)$, cílový stav $T \in S$
Output: vektor $\mathbf{x}(s) = P[s \models \Diamond\, T]$, pro každé $s \in S$

1 $S_0 \leftarrow \{s \in S \mid P[s \models \Diamond\, T\,] = 0\}$
2 $S_1 \leftarrow T$
3 $S_? \leftarrow S \setminus (S_0 \cup S_1)$
4 Vypočtení následující soustavy rovnic:

$$\mathbf{x}(s) = \begin{cases} 1 & : s \in S_1 \\ 0 & : s \in S_0 \\ \sum_{s' \in S} P(s, s') \cdot x(s') & : s \in S_? \end{cases}$$

5 return x

Provedením stejného výpočtu pro všechny cílové stavy t_x zjišťujeme, že pravděpodobnost dosažení všech cílových stavů je totožná. Tím pádem můžeme prohlásit, že je namodelovaná kostka férová.

Příklad 2.2.3 *Máme DTMC D z příkladu 2.1.2 a chceme ověřit zda platí $D \models P_{<0,17}[\Diamond t_1]$. Pro vyřešení tohoto problému musíme nejdříve vypočítat hodnotu $P[s_0 \models \Diamond\{t_1\}]$. Tu získáme pomocí algoritmu 1. V tomto případě dostaneme výsledek $P[s_0 \models \Diamond\{t_1\}] \cong 0,1\overline{6}$. Jelikož získaný výsledek splňuje požadavky: $0,1\overline{6} < 0,17$, docházíme k závěru $\mathcal{M} \models P_{<0,17}[\Diamond t_1]$. Pokud by vlastnost φ byla pozměněna a vypadala například takto $\varphi = P_{\geq 0.17}[\Diamond t_1]$, podmínka $0,1\overline{6} \geq 0,17$ by nebyla splněna a tedy by platilo $\mathcal{M} \not\models P_{\geq 0,17}[\Diamond\, t_1]$.*

2.2.3 Model checking pro MRM

Definice 2.2.4 ***(Očekávaná hodnota rewardu).*** *Očekávaná hodnota rewardu $ExpRew^{\mathcal{M}}(s_0 \models \Diamond T)$ značí průměrnou hodnotu součtu rewardů cesty π, při přechodu ze stavu s_0 do některého stavu z množiny cílových stavů $T \subseteq S$. V případě, kdy $P(s_0 \models \Diamond T) = 1$ se očekávaná hodnota rewardů získá vyřešením soustavy rovnic. Tyto rovnice se vytváří podle následujícího vzorce:*

$$r_s = \begin{cases} 0 & : s \in T \\ rew(s) + \sum_{s' \in S} P(s, s') \cdot r_{s'} & jinak \end{cases}$$

Hodnota r_s zde značí očekávanou hodnotu rewardu stavu s. V případě, kdy $P(s_0 \models \Diamond T) < 1$, platí $ExpRew^{\mathcal{M}}(s_0 \models \Diamond T) = \infty$.

Definice 2.2.5 ***(Reward vlastnost) [13].*** *Reward vlastnost $\varphi = R_{\bowtie\lambda}[\Diamond\, T]$, značí vlastnost MRM vzhledem k očekávané hodnotě rewardu při přechodu ze stavu $s_{init} \in S$ do některého stavu z množiny stavů $T \subseteq S$. Znak $\bowtie$ zde značí jednu z možností: $<, \leq, >, \geq$. Znak λ značí hraniční pravděpodobnost této vlastnosti, kdy $\lambda \in \mathbb{R}$. MRM $\mathcal{M}$ nabývá vlastnosti φ právě když platí $ExpRew^{\mathcal{M}}(s_0 \models \Diamond T) \bowtie \lambda$.*

Příklad 2.2.6 *Mějme MRM $\mathcal{M}$ z příkladu 2.1.5. Pro vypočtení $ExpRew^{\mathcal{M}}(s_0 \models \Diamond T)$, kde $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$, je potřeba podle vzorce v 2.2.4 vytvořit soustavu rovnic a následně z ní vypočítat požadovanou hodnotu. Vyřešením této soustavy rovnic získáme výsledek*

$ExpRew^{\mathcal{M}}(s_0 \models \Diamond T) = 3,\overline{6}$. *Z toho vyplývá, že namodelovaný model průměrně potřebuje* $3,\overline{6}$ *kroků k dosažení cíle.*

Příklad 2.2.7 *Mějme MRM* $\mathcal{M}$ *z příkladu 2.1.5. Ověřme zda platí* $\mathcal{M} \models R_{<3,7}[\Diamond\ T]$, *kde* T *obsahuje stavy definované ve stejném příkladu. Nejdříve musíme zjistit hodnotu* $ExpRew^{\mathcal{M}}(s_0 \models \Diamond T)$, *kterou získáme pomocí algoritmu na výpočet očekávané hodnoty rewardů 2.2.4. Získaná hodnota je v tomto případě* $ExpRew^{\mathcal{M}}(s_0 \models \Diamond T) = 3,\overline{6}$, *jak bylo vypočteno v příkladu 2.1.5. Jelikož* $3,\overline{6} \not\leq 3,7$ *docházíme k závěru* $\mathcal{M} \not\models R_{<3,7}[\Diamond\ T]$.

2.3 Protipříklady

Protipříklady [1] slouží ke zobecnění důvodu, proč daný DTMC nesplňuje specifikované požadavky. Protipříklady znázorňují určitou část Markovova řetězce, která sama o sobě odporuje požadované specifikaci. Hlavní využití protipříkladů je pro získání diagnostických informací o důvodu chyby pro její následné opravení. Protipříklady lze ale také využít pro zefektivnění pravděpodobnostní syntézy. Díky zobecnění důvodu nesplnění podmínky lze z prohledávané množiny řešení odstranit všechna řešení obsahující protipříklady a tak pravděpodobnostní syntézu zefektivnit.

Definice 2.3.1 ***(Protipříklad pro DTMC) [13].*** *Nechť* $D = (S, s_{init}, P)$ *je DTMC,* $T \subseteq S$ *je množina cílových stavů a* $\mathbb{P}_{\bowtie\lambda}[\Diamond T]$ *je pravděpodobnostní vlastnost, kterou nesplňuje DTMC* D. *Pokud podmnožina* $S' \subseteq S$ *obsahuje stav* $s_{init} \in S$, *nazýváme ji selekcí. Indukcí* D *selekcí* S' *vzniká podsystém* $D' = (S' \uplus \{t\}, s_{init}, P')$ *ve kterém je nový stav* $t \notin S$ *a* P' *je definováno následovně:*

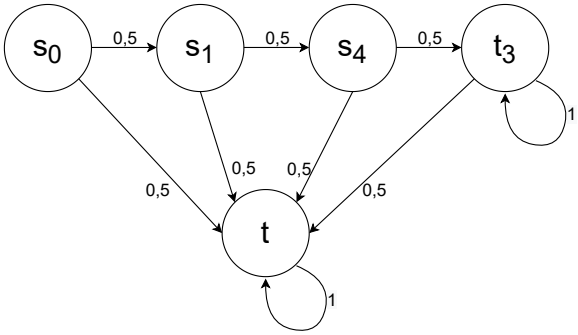
$$P'(s,s') = \begin{cases} P'(s,s') & : s,s' \in S' \\ \sum_{s'' \in S \setminus S'} P(s,s'') & : s \in S' \ a\ s' = t \\ 1 & : s = s' = t \\ 0 & jinak \end{cases}$$

Selekce S' *je protipříkladem diskrétního Markovova řetězce* D *pro vlastnost* $\mathbb{P}_{\lhd\lambda}(\Diamond T)$ *právě tehdy, když* $\mathbb{P}_{\bowtie\lambda}(\Diamond T')$ *odporuje podsystému* S' *kde* $T' = T \cap S'$.

Příklad 2.3.2 *Mějme DTMC* $D = (S, s_{init}, P)$ *z příkladu 2.1.2. Tentokrát chceme vytvořit protipříklad pro safety vlastnost* $\varphi = P_{\leq 0.1}[\Diamond\ t_3]$. *Zvolíme-li si množinu kritických stavů* $C = \{s_0, s_1, s_4, t_3\}$, *tak výsledný podsystém* $D \downarrow C$ *je kritickým podsystémem diskrétního Markovova řetězce* D. *Získaný kritický podsystém je znázorněn v grafu 2.2. V grafu si můžeme všimnou, že zde už není stav* s_2, *protože* $s_2 \notin C$. *Tento stav je zde nahrazen stavem* t, *který je absorbující a je zde aby platilo pravidlo, že součet pravděpodobností přechodů do všech stavů je u každého stavu roven jedné. Také si můžeme ověřit, že pravděpodobnost* $P[s_0 \models \Diamond\{t_3\}] = 0,125$, *tedy přesahuje požadovanou velikost podmínky* φ.

2.4 Rodiny Markovových řetězců

Doposud jsem se věnoval konkrétním Markovovým řetězcům a jejich model checkingu. Pro účely pravděpodobnostní syntézy je výhodné zápis velké množiny Markovových řetězců zjednodušit pomocí použití Markovových rodin. Ty se skládají z modelu podobnému klasickému DTMC ve kterém jsou místo některých stavů parametry, které mohou symbolizovat



Obrázek 2.2: Příklad 2: ukázka kritického podsystému $D \downarrow C$ pro safety vlastnost

libovolný stav z předem definované množiny stavů, která je tomuto parametru přiřazena. Výsledná rodina Markovových řetězců reprezentuje množinu Markovových řetězců pro každou kombinaci hodnot parametrů.

Definice 2.4.1 ***(Rodina DTMC) [14].*** *Rodina Markovových řetězců je uspořádaná čtveřice $\mathcal{D} = (S, s_{init}, K, \mathcal{B})$ kde S a s_{init} odpovídají definici 2.1.1, K je konečná množina parametrů $k \in K$ pro které platí $T_k \subseteq S$ a $\mathcal{B} : S \to Distr(K)$ je funkcí pro rozdělení pravděpodobnosti přechodů.*

Na rozdíl od DTMC který obsahuje funkci $\mathcal{P}$ určující pravděpodobnostní rozdělení přechodů do následujících stavů, rodina Markovových řetězců obsahuje funkci $\mathcal{B}$. Ta určuje pravděpodobnostní rozdělení přechodů do následujících stavů obdobně jako funkce $\mathcal{P}$. Navíc ale umožňuje nahrazení některých cílových stavů parametry $k \in K$. Každý parametr k obsahuje množinu stavů $s \in S$, které zastupuje. Přidělením konkrétních stavů jednotlivým parametrům vznikají konkrétní DTMC, které jsou členy dané rodiny DTMC.

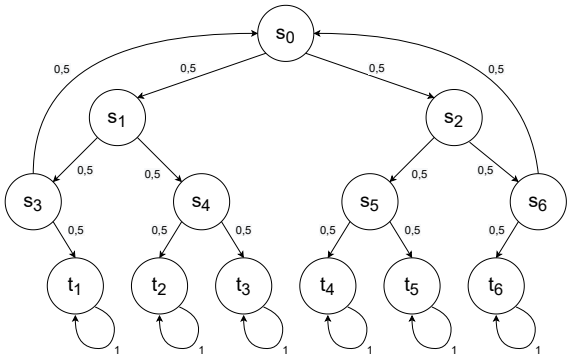
Definice 2.4.2 ***(Realizace rodiny DTMC) [14].*** *Realizace rodiny Markovových řetězců $\mathcal{D} = (S, s_{init}, K, \mathcal{B})$ je funkce $r : K \to S$ kde $\forall k \in K : r(k) \in T_k$. Realizací r se vytváří MC $D_r := (S, s_{init}, \mathcal{B}(r))$, kde je $\mathcal{B}(r)$ maticí pravděpodobností přechodů ve které je každé $k \in K$ v $\mathcal{B}$ nahrazeno $r(k)$. Množinu všech realizací rodiny $\mathcal{D}$ budeme dále označovat $\mathcal{R}^{\mathcal{D}}$.*

Počet realizací rodiny Markovových řetězců odpovídá součinu počtu prvků všech parametrů $k \in K$. Tedy platí $|\mathcal{R}^{\mathcal{D}}| = \prod_{k \in K} |T_K|$. Z toho vyplývá, že počet realizací rodiny Markovových řetězců roste exponenciálně s přibývajícím počtem parametrů K.

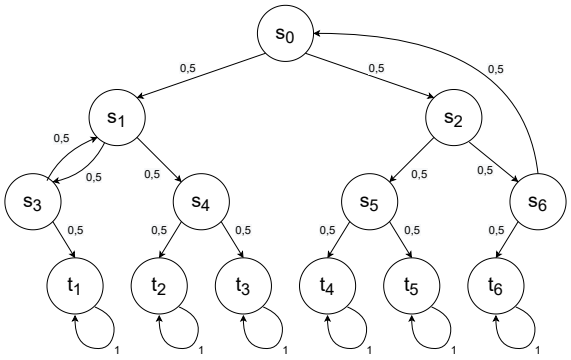
Příklad 2.4.3 *Mějme rodinu Markovových řetězců $\mathcal{D} = (S, s_{init}, K, \mathcal{B})$ kde S a s_{init} odpovídají S a s_{init} z příkladu 2.1.2, $K = \{X, Y, k_{s_n}, k_{t_n}\}$ kde $T_X = \{s_0, s_2\}$, $T_Y = \{s_0, s_1\}$ a k_{s_n} značí skupinu parametrů pro něž platí $T_{k_{s_n}} = \{s_n\}$ a nápodobně $T_{k_{t_n}} = \{t_n\}$. Funkce*

pro rozdělení pravděpodobnosti přechodů $\mathcal{B}$ vypadá následovně:

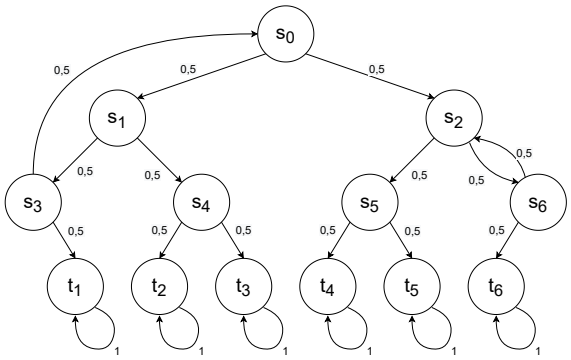
$$\begin{aligned}
\mathcal{B}(s_0) &= 0,5 : k_{s_1} + 0,5 : k_{s_2},\\
\mathcal{B}(s_1) &= 0,5 : k_{s_3} + 0,5 : k_{s_4},\\
\mathcal{B}(s_2) &= 0,5 : k_{s_5} + 0,5 : k_{s_6},\\
\mathcal{B}(s_3) &= 0,5 : Y + 0,5 : k_{t_1},\\
\mathcal{B}(s_4) &= 0,5 : k_{t_2} + 0,5 : k_{t_3},\\
\mathcal{B}(s_5) &= 0,5 : k_{t_4} + 0,5 : k_{s_5},\\
\mathcal{B}(s_6) &= 0,5 : X + 0,5 : k_{t_6},\\
\mathcal{B}(t_n) &= 1 : k_{t_n}.
\end{aligned}$$



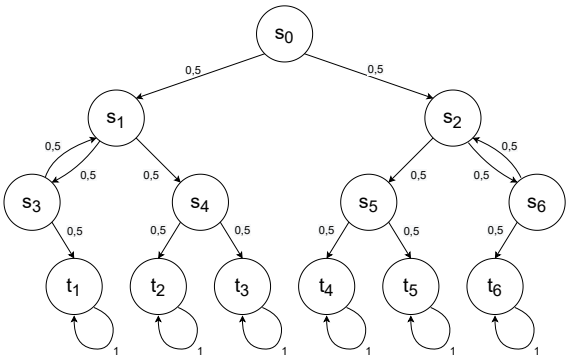
(a) $X = s_0, Y = s_0$



(b) $X = s_0, Y = s_1$



(c) $X = s_2, Y = s_0$



(d) $X = s_2, Y = s_1$

Obrázek 2.3: Ukázka modelů jednotlivých realizací rodiny Markovových řetězců $\mathcal{M}$ z příkladu 2.4.3

Grafy všech čtyř realizací dané rodiny Markovových řetězců jsou znázorněny v obrázku 2.3. Na tomto příkladu můžeme všimnout, že počet realizací odpovídá $\prod_{k \in K} |T_K| = |T_X| \cdot |T_Y| \cdot |T_{k_{s_x}}| \cdot |T_{k_{s_x}}| = 2 \cdot 2 \cdot 1 \cdot 1 = 4$. *Všimněme si jak změny parametrů ovlivnily konkrétní realizace rodiny Markovových řetězců. Také si všimněme, že realizace 2.3d odpovídá modelu popisovaném v příkladu 2.1.2 a tedy lze použít pro modelování simulace hodu kostkou pomocí série hodů mincí. Po zkoumání ostatních realizací můžeme dojít k závěru, že k modelování férové kostky by mohl sloužit i model 2.3a. Tento model by ale k výsledku potřeboval průměrně více kroků.*

2.5 Markovovy rozhodovací procesy

Jedním ze způsobů, jak uvažovat o rodinách Markovových řetězců je použití Markovových rozhodovacích procesů. Markovovy rozhodovací procesy [15, 2] jsou rozšířením DTMC, kdy každému stavu může příslušet více akcí, které rozhodují o rozvržení pravděpodobnosti přechodů z tohoto stavu.

Definice 2.5.1 *[15] Markovův rozhodovací proces je uspořádaná čtveřice* $M = (S, s_{init}, Act, \mathcal{P})$*, kde* S *a* s_{init} *odpovídají definici 2.1.1,* Act *je konečná množina akcí a* $\mathcal{P} : S \times Act \nrightarrow Distr(S)$*. Pro všechny možné akce stavu* $s \in S$*, jsou označeny* $Act(s)$*. Platí že* $Act(s) = \{a \in Act | \mathcal{P}(s, a) \neq \bot$.

Při použití akce $a \in Act$ ve stavu $s \in S$ se rozložení pravděpodobnosti přechodů do ostatních stavů značí $\mathcal{P}(s)(a)$. Pravděpodobnost přechodu do konkrétního stavu $s' \in S$ se značí $\mathcal{P}(s)(a)(s')$. Výběr akce $a \in Act(s)$ pro rozdělení pravděpodobností přechodů ze stavu s probíhá nedeterministickým způsobem. Cesta MDP M je posloupnost $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots s_{n-1} \xrightarrow{a_{n-1}} s_n$ kde pro každý stav $s_i \in \pi$ kde $i > 0$ platí $\mathcal{P}(s_{i-1})(a)(s_i) > 0$. Pravděpodobnost cesty π se vypočítá pomocí součinu pravděpodobností všech přechodů mezi stavy dané cesty za použití příslušných akcí. Tedy $P(\pi) = \prod_{i>0} \mathcal{P}(s_{i-1})(a)(s_i)$. Případné rozvržení stavových rewardů určuje funkce $rew : S \to \mathbb{R}_{\geqslant 0}$. Získání rewardu $rew(s)$ probíhá při odchodu ze stavu s. Množinu všech nekonečných cest ze stavu $s \in S$ budeme dále označovat $Paths^M(s)$, množinu konečných cest $Paths^M_{fin}(s)$ a poslední stav cesty $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n$ budeme značit $last(\pi)$. Strategie (scheduler) σ je funkce umožňující plánování použití příslušných akcí v jednotlivých stavech, tak by rozhodování bylo deterministické. Po příchodu do stavu s je strategií σ vybrána vhodná akce $a \in Act(s)$ a rozdělení pravděpodobnosti přechodů je určeno pomocí $\mathcal{P}(s)(a)$[1].

Definice 2.5.2 *[15] Strategie pro MDP* $M = (S, s_0, Sct, \mathcal{P})$ *je funkce* $\sigma : Paths^M_{fin} \to Act$ *pro kterou platí* $\sigma(\pi) \in Act(last(\pi))$ *pro všechna* $\pi \in Paths^M_{fin}$*. Strategie jsou bezpaměťové pokud* $last(\pi) = last(\pi) \Longrightarrow \sigma(\pi) = \sigma(\pi')$ *pro všechna* $\pi, \pi' \in Paths^M_{fin}$.

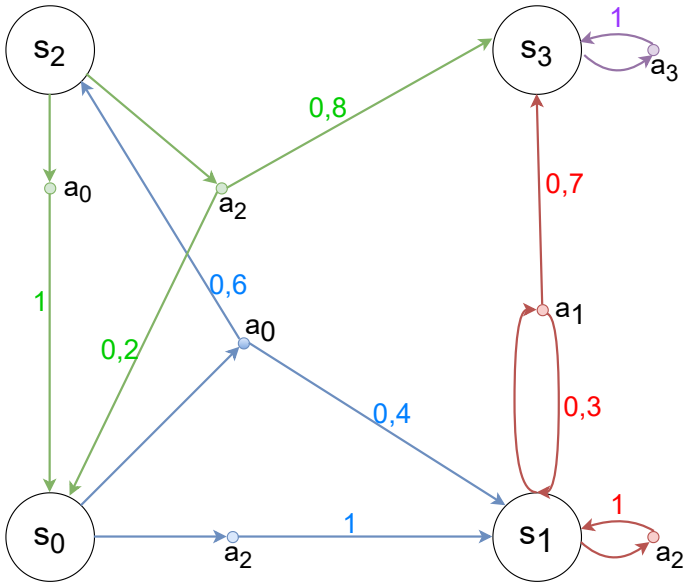
Definice 2.5.3 *[2] Mějme MDP* $M = (S, s_{init}, Act, \mathcal{P})$*. Strategie* $\sigma \in \sum^M$ *indukuje DTMC* $M^\sigma = (Paths^M_{fin}, s_{init}, \mathbf{P}^\sigma)$*, právě když* $\mathbf{P}^\sigma(\pi, \pi \xrightarrow{\sigma(\pi)} s') = \mathcal{P}(last(\pi), \sigma(\pi), s')$ *a v ostatních případech platí* $\mathbf{P}^\sigma(\cdot, \cdot) = 0$.

Jestliže máme MDP $M = (S, s_{init}, Act, \mathcal{P})$ a bezpaměťovou strategii $\sigma \in \sum^M$, tak DTMC M^σ vytvořený indukcí M a σ odpovídá DTMC $M'(S, s_{init}, \mathbf{P}^\sigma)$ kde $\mathbf{P}^\sigma \equiv \mathcal{P}(s, \sigma(s))$.

Příklad 2.5.4 *Mějme MDP* $M = (S, s_{init}, Act, \mathcal{P})$ *kde* $S = \{s_0, s_1, s_2, s_3\}$*,* $s_{init} = s_0$*,* $Act = \{a_0, a_1, a_2, a_3\}$ *a* $\mathcal{P}$ *odvozeným z grafu 2.4. V tomto grafu z každého stavu* s_i *vedou nejdříve šipky k akci* a_n*, ze které pak následuje rozdělení přechodů do konkrétních stavů. Toto rozdělení odpovídá* $\mathcal{P}(s_i)(a_n)$*. Například v počátečním stavu* s_0 *je možné provést akce* $Act(s_0) = \{a_0, a_2\}$*. Při použití akce* a_0 *ve stavu* s_0 *je pravděpodobnost přechodu do stavu* s_2 *rovna* $0,6$ *a pravděpodobnost přechodu do stavu* s_1 *je* $0,4$*. Tedy* $\mathcal{P}(s_0)(a_0)(s_2) = 0,6$ *a* $\mathcal{P}(s_0)(a_0)(s_1) = 0,4$*. Použití akcí* a_1 *a* a_3 *ve stavu* s_0 *není možné, tedy platí* $\mathcal{P}(s_0)(a_1) = \bot$ *a* $\mathcal{P}(s_0)(a_3) = \bot$.

[1] Zde uvažujeme pouze situaci, kdy je strategie bezpaměťová

Mějme cestu $\pi = s_0 \xrightarrow{a_0} s_2 \xrightarrow{a_0} s_0 \xrightarrow{a_2} s_1 \xrightarrow{a_1} s_1 \xrightarrow{a_1} s_3$. *Pravděpodobnost této cesty je* $P[\pi] = \mathcal{P}(s_0)(a_0)(s_2) \cdot \mathcal{P}(s_2)(a_0)(s_0) \cdot \mathcal{P}(s_0)(a_2)(s_1) \cdot \mathcal{P}(s_1)(a_1)(s_1) \cdot \mathcal{P}(s_1)(a_1)(s_3) = 0,6 \cdot 1 \cdot 1 \cdot 0,3 \cdot 0,7 = \frac{63}{500}$.



Obrázek 2.4: graf MDP

2.5.1 Model checking pro MDP

Cílem model checkingu pro MDP [6] je zjistit, zda pro všechny realizace daného MDP $\mathcal{M} = (S, s_{init}, Act, \mathcal{P})$ platí zadaná vlastnost φ. Tedy platí $\mathcal{M} \models \varphi \Leftrightarrow \forall \sigma \in \sum^{\mathcal{M}} : \mathcal{M}^{\sigma} \models \varphi$. Tato práce se zabývá pravděpodobnostními vlastnostmi ve tvaru $\varphi = P_{\bowtie\lambda}[\Diamond\ T]$ a reward vlastnostmi ve tvaru $\varphi = R_{\bowtie\lambda}[\Diamond\ T]$ definovanými v kapitole 2.2. Zda všechny realizace daného mdp odpovídají specifikaci se pro safety vlastnosti zjišťuje pomocí nalezení realizace σ_{max}. Pro realizaci σ_{max} platí $\forall \sigma \in \sum^{\mathcal{M}} : \mathbb{P}[\mathcal{M}^{\sigma} \models \varphi] \leq \mathbb{P}[\mathcal{M}^{\sigma_{max}} \models \varphi]$. Tato vlastnost realizace σ_{max} zajišťuje, že pokud platí $\mathcal{M}^{\sigma_{max}} \models \varphi$, tak jednoznačně platí i $\mathcal{M} \models \varphi$. To zaručuje fakt, že pokud realizace s nejvyšší možnou pravděpodobností splňuje safety vlastnost φ, tak vlastnost φ musí splňovat všechny realizace, jelikož pravděpodobnost σ_{max} nemohou překonat. U liveness vlastností se zda platí $\mathcal{M} \models \varphi$ zjišťuje obdobným způsobem. Místo použití realizace σ_{max} se ale využívá realizace σ_{min}, pro kterou platí $\forall \sigma \in \sum^{\mathcal{M}} : \mathbb{P}[\mathcal{M}^{\sigma_{min}} \models \varphi] \leq \mathbb{P}[\mathcal{M}^{\sigma} \models \varphi]$. Hodnotu σ_{max} lze získat minimalizováním $\sum_{s \in S} x_s$ pro následující soustavu nerovnic pomocí lineárního programování.

$$
\begin{aligned}
x_s &= 1 && : \forall s \ \in S^1_{max} \\
x_s &= 0 && : \forall s \ \in S^0_{max} \\
x_s &\leq \textstyle\sum_{s' \in S} P(s)(a)(s') \cdot x(s') && : \forall s \ \notin (S^0_{max} \cup S^0_{max}) a \in Act(s)
\end{aligned}
$$

Množina S^1_{max} zde obsahuje stavy pro které platí $s \in S | \exists \sigma \in \sum^{\mathcal{M}} : \mathbb{P}[\mathcal{M}^{\sigma}, s \models \varphi] = 1$ a množina S^0_{max} obsahuje stavy pro které platí $s \in S | \forall \sigma \in \sum^{\mathcal{M}} : \mathbb{P}[\mathcal{M}^{\sigma}, s \models \varphi] = 0$.

Hodnotu σ_{min} lze získat obdobným způsobem. Tentokrát je potřeba pomocí lineárního programování maximalizovat $\sum_{s \in S} x_s$ pro :

$$\begin{array}{ll} x_s = 1 & : \forall s \ \in S^1_{min} \\ x_s = 0 & : \forall s \ \in S^0_{min} \\ x_s \geq \sum_{s' \in S} P(s)(a)(s') \cdot x(s') & : \forall s \ \notin (S^0_{min} \cup S^0_{min}) a \in Act(s) \end{array}$$

Množina S^0_{min} zde obsahuje stavy $s \in S | \exists \sigma \in \sum^{\mathcal{M}} : \mathbb{P}[\mathcal{M}^\sigma, s \models \varphi] = 0$ a množina S^1_{min} stavy $s \in S | \forall \sigma \in \sum^{\mathcal{M}} : \mathbb{P}[\mathcal{M}^\sigma, s \models \varphi] = 1$.

2.6 Syntéza pravděpodobnostních programů

Syntéza pravděpodobnostních programů se zabývá automatickým návrhem pravděpodobnostního programu, který splňuje specifikované požadavky. V této práci se věnuji typu pravděpodobnostní syntézy, kdy máme program sketch který znázorňuje program s možností různé volby parametrů. Cílem syntézy je v tomto případě nalezení takového nastavení daných parametrů, aby výsledný pravděpodobnostní program splňoval všechny požadované specifikace. Feasibility syntéza slouží k vyhledání z množiny možných řešení jakéhokoliv řešení které splňuje požadovanou specifikaci. Optimální syntéza vyhledává řešení které dané specifikace splňuje nejvíce.

Definice 2.6.1 ***(Feasibility syntéza) [14].*** *Nechť $\mathcal{D}$ je rodina Markovových řetězců a φ je pravděpodobnostní vlastnost nebo reward vlastnost. Cílem feasibility syntézy pravděpodobnostních programů je najít takovou realizaci $r \in R^{\mathcal{D}}$ aby $D_r \models \varphi$.*

Definice 2.6.2 ***(Optimální syntéza) [14].*** *Nechť $\mathcal{D}$ je rodina Markovových řetězců a φ je pravděpodobnostní vlastnost nebo reward vlastnost. Cílem optimální syntézy pravděpodobnostních programů je najít $r^* \in R^{\mathcal{D}}$ aby $r^* \models argmax_{r \in R^{\mathcal{D}}} P[D_r \models \Diamond T]$.*

Pokud v následujícím textu není specifikováno o který typ pravděpodobnostní syntézy se jedná, tak budeme uvažovat feasibility syntézu.

Nejzákladnějším způsobem syntézy pravděpodobnostních programů je takzvaná metoda one-by-one [4]. Principem této metody je postupné procházení všech realizací dané rodiny Markovových řetězců a kontrola zda konkrétní realizace splňují specifikovanou vlastnost. V případě, kdy kontrolovaná realizace požadovanou vlastnost splňuje, je tato realizace vrácena jako výsledek syntézy a syntéza se ukončí. Pokud kontrolovaná realizace vlastnost nesplňuje, je vyřazena z množiny kontrolovaných realizací a syntéza pokračuje kontrolou následující realizace. Tato metoda je časově velmi náročná, protože se pro kontrolu celé rodiny Markovových řetězců musí zkontrolovat všechny jednotlivé realizace této rodiny Markovových řetězců. Jelikož počet realizací roste exponenciálně s přibývajícím počtem parametrů, počet realizací rozsáhlých rodin Markovových řetězců je příliš velký a tato metoda je pro takto rozsáhlé rodiny nevhodná.

Příklad 2.6.3 *Mějme rodinu Markovových řetězců $\mathcal{D}$ z příkladu 2.4.3 a pravděpodobnostní vlastnost $\varphi = P_{\leq 0.15}[\Diamond \ s_1]$. Výsledkem syntézy rodiny $\mathcal{D}$ pro specifikaci φ je realizace $r \in$*

$R^{\mathcal{D}}$ jejíž parametry jsou $X = s_2$ a $Y = s_0$ 2.3c. Pro D_r této realizace platí $P[s_{init} \models \Diamond\{s_1\}] = 0,1429$ a jelikož platí $0,1429 \geq 0,15$, tato realizace splňuje specifikaci φ. Pro ostatní realizace platí $P[s_{init} \models \Diamond\{s_1\}] \geq 0,15$ a tedy nevyhovují specifikované podmínce.

Problém vysoké náročnosti metody one-by-one je možné redukovat pomocí použití evolučních vyhledávacích algoritmů [7]. Velkou nevýhodou této metody je, že není úplná. To znamená, že nezaručuje kontrolu všech realizací. Může tedy vrátit výslednou informaci o nenalezení žádné realizace odpovídající požadované specifikaci i v případě, že rodina Markovových řetězců takovou realizaci obsahuje. Touto skupinou metod se z tohoto důvodu tato práce zabývat nebude.

2.6.1 Protipříklady řízená induktivní syntéza (CEGIS)

Jednou z dalších možností řešení syntézy pravděpodobnostních programů jsou metody založené na protipříklady řízené induktivní syntéze (dále už pouze CEGIS) [14]. Velikou výhodou naproti metodám založeným na evolučních vyhledávacích algoritmech je, že metody CEGIS jsou úplné. Také jsou výrazně efektivnější než metoda one-by-one.

Průběh metod CEGIS je podobný metodě one-by-one, ale využívá se zde zobecnění důvodu nevyhovění specifikací. Postupně se prochází jednotlivé realizace r rodiny Markovových řetězců $\mathcal{D} = (S, s_{init}, K, \mathcal{B})$ a kontroluje se zda odpovídají specifikovaným vlastnostem φ. Pokud $r \models \varphi$, je r vrácena jako výsledek syntézy, stejně jako je to u metody one-by-one. Při nalezení každé realizace $r \not\models \varphi$, je pro danou realizaci vytvořen protipříklad. Analýzou vytvořeného protipříkladu se zjistí, které parametry z množiny K mají vliv na chování daného protipříkladu a které jsou naopak pro jeho chování irelevantní. Parametry které chování protipříkladu ovlivňují budeme dále označovat $\overline{K}$. Následně se z množiny prohledávaných realizací odstraní všechny realizace, které obsahují stejné parametry $\overline{K}$ jako daný protipříklad a tedy je zaručeno, že nevyhovují požadovaným specifikacím.

Algoritmus 2: CEGIS

```
1  CEGIS(D = (S, s_init, K, B), φ)
2  ψ ← Initialise(D)
3  R ← GetRealization(ψ)
4  while R ≠ Unsat do
5  |  C ← Verify(D, R, φ)
6  |  if C then
7  |  |  return R
8  |  end if
9  |  ψ ← ψ ∧ (⋀_{R̄∈C} LearnFromConflict(D, R̄))
10 |  R ← GetRealization(ψ)
11 end while
12 return Unsat
```

Přesnější postup metod CEGIS je popsaný algoritmem 2. Funkce `Inicialise()` zde z rodiny Markovových řetězců, která je na vstupu, vytvoří množinu realizací, které se dále budou analyzovat. Funkce `GetRealization()` vrací jednu realizaci z množiny prohledávaných realizací. Funkce `Verify()` kontroluje, zda realizace R rodiny $\mathcal{D}$ odpovídá specifikaci φ. Funkce `LearnFromConflict()` analyzuje konkrétní realizaci a slouží k nalezení ostatních realizací, které obsahují stejný protipříklad.

2.6.2 Metody založené na zjemňování abstrakce (CEGAR)

Dalším možným způsobem řešení pravděpodobnostní syntézy je použití metod založených na zjemňování abstrakce (dále už pouze CEGAR) [15]. Principem těchto metod je vyhodnocení rodiny Markovových řetězců jako celku pomocí model checkingu MDP. MDP umožňuje nadaproximaci rodiny DTMC. Analýzou takového MDP je možné zjistit případy, kdy danou podmínku splňují všechny realizace, nebo ji naopak nesplňuje žádná z realizací rodiny Markovových řetězců. Pomocí těchto informací v kombinaci s dělením množiny možných realizací do podmnožin lze pravděpodobností syntéza velice zefektivnit.

Přesnější postup metod CEGAR je popsán algoritmem 3 [2], který je konkrétně určený pro syntézu se safety vlastností. Vstupem algoritmu je rodina Markovových řetězců $\mathcal{D} = (S, s_{init}, K, \mathcal{B})$, a safety vlastnost $\varphi \equiv \mathbb{P}_{\leq\lambda}[\Diamond T]$. $\mathcal{M}^{\mathcal{D}}$ je `quotientMDP` pro rodinu MC $\mathcal{D}$, kde platí MDP $\mathcal{M}^{\mathcal{D}} = (S, s_{init}, \mathcal{R}^{\mathcal{D}}, \mathcal{P})$, kde $\mathcal{P}(\cdot)(r) \equiv \mathcal{B}_r$. Množina U zde obsahuje všechny různé množiny realizací $\mathcal{R}$, které je potřeba analyzovat. Po inicializaci množina U obsahuje pouze jednu množinu realizací a to $\mathcal{R}^{\mathcal{D}}$. Poté následuje postupné procházení všech $\mathcal{R} \in U$ a jejich následná analýza. $\mathcal{M}^{\mathcal{D}}[\mathcal{R}]$ značí $\mathcal{M}^{\mathcal{D}[\mathcal{R}]}$ kde $\mathcal{R} \subseteq \mathcal{R}^{\mathcal{D}}$. Analýzou $\mathcal{M}^{\mathcal{D}}[\mathcal{R}]$ se získají hodnoty x_{max}, σ_{max}, x_{min} a σ_{min}, jak je popsáno v 2.5.1. V případě kdy $x_{min}(s_{init}) > \lambda$ víme, že žádná z realizací nesplňuje specifikace, tedy se může rovnou přejít k analýze jiné množiny realizací $\mathcal{R} \in U$. Pokud už žádná $\mathcal{R} \in U$ neexistuje, neexistuje žádná realizace $r \in R^{\mathcal{D}}$ splňující φ. V případě kdy $x_{max}(s_{init}) \leq \lambda$ víme, že pro všechna $r \in \mathcal{R}$ platí $\mathcal{D}_r \models \lambda$. V tomto případě může být výsledkem syntézy libovolné $r \in \mathcal{R}$. Pokud ale platí $x_{min}(s_{init}) \leq \lambda \leq x_{max}(s_{init})$, mohou vlastnost φ splňovat pouze některé realizace z množiny $\mathcal{R}$. V takovém případě se množina $\mathcal{R}$ rozdělí na 2 podmnožiny, které se vloží do množiny U a pokračuje se analýzou následující množiny $\mathcal{R} \in U$.

Algoritmus 3: algoritmus pro CEGAR

Input: Rodina Markovových řetězců $\mathcal{D} = (S, s_{init}, K, \mathcal{B})$, safety vlastnost $\varphi \equiv \mathbb{P}_{\leq\lambda}[\Diamond T]$

Output: Realizace $r \in R^{\mathcal{D}}$, pro kterou platí $\mathcal{D}_r \models \lambda$, nebo v případě že žádná taková realizace není Unsat

```
M^D ← buildQuotientMDP(D)
U ← {R^D}
while U ≠ ∅ do
    select R ∈ U and U ← U \R
    (x_max, σ_max) ← solveMaxMDP(M^D[R], T)
    (x_min, σ_min) ← solveMinMDP(M^D[R], T)
    if x_min(s_init) > λ then
        continue
    end if
    if x_max(s_init) ≤ λ then
        return any(R)
    end if
    (R_⊤, R_⊥) ← split(R)
    U ← U ∪ {R_⊤, R_⊥}
end while
return Unsat
```

2.6.3 Hybridní metoda pravděpodobnostní syntézy

Dalším způsobem řešení pravděpodobnostní syntézy je použití nové metody [3], která současně využívá metody CEGAR i CEGIS. Při tvorbě protipříkladů metodou CEGIS se zde využívají informace o minimální a maximální pravděpodobnosti dosažení cílových stavů, získané zjemňováním abstrakce. Díky tomu je možné vytvářet menší a obecnější protipříklady a tedy pravděpodobnostní syntézu provádět efektivněji.

Při tvorbě protipříkladu se v této metodě používají `přesměrované` Markovovy řetězce.

Definice 2.6.4 ***(Přesměrované DTMC) [3].*** *Nechť $D = (S, s_{init}, \mathbf{P})$ je DTMC a stavy $s_\top, s_\bot \notin S$. Mějme množinu expandovaných stavů $C \subseteq S$ a funkci $\gamma : S \backslash C \to [0, 1]$. Přesměrováním Markovova řetězce D použitím množiny stavů C a funkce γ vzniká DTMC $D \downarrow C[\gamma] = (S \cup \{s_\bot, s_\top\}, s_{init}, \mathbf{P}^C_\gamma)$, kde $\mathbf{P}^C_\gamma$ je definováno následovně:*

$$\mathbf{P}^C_\gamma(s) = \begin{cases} \mathbf{P}(s) & : s \in C \\ [s_\top \to \gamma(s), s_\bot \to (1 - \gamma(s))] & : s \in S \backslash C \\ [s \to 1] & : s \in \{s_\top, s_\bot\} \end{cases}$$

V případě této metody, vrací funkce γ příslušné hodnoty $lb^{\mathcal{R}}(s)$ nebo $ub^{\mathcal{R}}(s)$. Tyto hodnoty odpovídají hodnotám $x_{min}(s)$ a $x_{max}(s)$ definovaným v kapitole 2.6.2. Hodnota $lb^{\mathcal{R}}(s)$ tedy značí nejmenší možnou pravděpodobnost dosažení stavu množiny cílových stavů ze stavu s pro všechny realizace $r \in R$. Hodnota $ub^{\mathcal{R}}(s)$ naopak značí pravděpodobnost největší. Neboli platí: $lb^{\mathcal{R}}(s) \leq \mathbb{P}[\mathcal{D}_r, s \models \Diamond T] \leq ub^{\mathcal{R}}(s)$. Stav $s_\top$ se zde přidává do množiny cílových stavů. Přechod $s \xrightarrow{\gamma(s)} s_\top$ se stává takzvanou zkratkou. Pomocí této zkratky se danému stavu přiřazuje nejmenší nebo největší možná pravděpodobnost dosažení cílového stavu. Smysl použití zkratek je vysvětlen v následujícím příkladu.

Příklad 2.6.5 *Mějme rodinu Markovových řetězců $\mathcal{D}$ odpovídající realizaci 2.3d z příkladu 2.4.3 a pravděpodobnostní vlastnost $\varphi = P_{\geq 0.15}[\Diamond\ t_3]$. Pokud bychom tvořili klasický protipříklad, tak jak je to popsáno v sekci 2.3, tak by množinou kritických stavů byla $C = \{s_0, s_1, s_3, s_4, t_3\}$. Po pozorném zkoumání všech realizací si ale můžeme všimnout, že by dostačující protipříklad tvořila i výrazně menší množina kritických stavů $C' = \{s_0, s_1, s_3\}$,jelikož všechny realizace které mají v těchto stavech stejné chování porušují specifikovanou podmínku. Tento protipříklad ale klasickým způsobem není možné provést, protože nevede do cílového stavu a tím pádem požadovanou specifikaci sám o sobě neporušuje. Díky použití výše definovaných zkratek to ale možné je a tím pádem je možné tvořit výrazně menší protipříklady.*

Kapitola 3

Rozšíření specifikačního jazyka pravděpodobnostní syntézy

Tato kapitola pojednává o mém návrhu a implementaci rozšíření metody syntézy pravděpodobnostních programů využívající kombinace induktivního a deduktivního přístupu. První rozšíření umožňuje při pravděpodobnostní syntéze použít reward vlastnosti. Druhé rozšíření probírané v této kapitole umožňuje využití until vlastností.

3.1 Reward vlastnosti

V této sekci se budu zabývat hlavní částí této práce, kterou je rozšíření specifikačního jazyka metody pravděpodobnostní syntézy o možnost použití rewardů. Toto rozšíření umožňuje jednotlivým stavům navrhovaného modelu přiřadit konkrétní hodnoty ceny. Tyto hodnoty nazýváme rewardy. Také toto rozšíření umožňuje při syntéze specifikovat podmínky týkající se rewardů. Díky tomu lze vyhledávání řešení syntézy omezit na základě očekávaného počtu získaných rewardů do dosažení cílového stavu. To lze využít například při hledání řešení na základě průměrného počtu kroků k jeho vyřešení, nebo s omezenou spotřebou zdrojů.

První část této kapitoly se věnuje klasickému způsobu tvorby protipříkladů pro modely s rewardy. Dále se zde budeme zabývat způsobem, jak využít znalosti metody pro pravděpodobnostní syntézu kombinující induktivní a deduktivní přístup k tvorbě protipříkladů pro modely pracující s rewardy.

3.1.1 Standardní tvorba protipříkladů pro rewardy

Nejdříve si popíšeme tvorbu protipříkladů pro reward vlastnosti bez využití informací získaných pomocí zjemňování abstrakce. Principem těchto protipříkladů je nalezení takového podsystému, který obsahuje počáteční stav realizace a zároveň nesplňuje požadovanou realizaci. Při pravděpodobnostní syntéze využíváme tvorbu protipříkladů k zobecnění důvodu nesplnění požadavku. Díky tomuto zobecnění je možné z prohledávané množiny vždy odstranit všechny realizace, které daný protipříklad obsahují. Z toho plyne, že je naším cílem zajistit, aby vytvořené protipříklady byly co nejmenší. Čím méně parametry je chování daného protipříkladu ovlivněno, tím je daný protipříklad obecnější a je díky němu možné odstranit více realizací z množiny prohledávaných řešení.

Definice 3.1.1 ***(Protipříklad pro MRM) [13].*** *Nechť $\mathcal{M} = (D, rew)$ je MRM kde $D = (S, s_{init}, P)$. Selekcí stavů $S' \subseteq S$ modelu $\mathcal{M}$ vzniká MRM $\mathcal{M}' = (D', rew')$, kde platí*

$D' = (S' \uplus \{t\}, s_{init}, P')$, *kde* $t \notin S$ *je nový stav,* $rew'(t) = 0$, $rew'(s) = rew(s)$ *pro všechna* $s \in S'$ *a* P' *odpovídá definici 2.3.1. Podsystém* $\mathcal{M}'$ *je protipříkladem vůči vlastnosti* $R_{\lhd\lambda}(\Diamond T)$ *kde* $\lhd \in \{<, \leq\}$, *právě tehdy když* $\mathcal{M}' \not\models R_{\lhd\lambda}(\Diamond T')$ *kde* $T' = (T \cap S') \cup \{t\}$.

Jedná se o rozšíření protipříkladu pro pravděpodobnostní vlastnosti v tom, že je zde potřeba definovat nový absorbující stav t pro který platí $rew(t) = 0$. Tím se zaručí, že daný podsystém v přidaném cílovém stavu t nezíská žádné nové rewardy, které by narušili správné chování podsystému v kontextu použití jako protipříkladu.

Příklad 3.1.2 *Mějme MRM* $\mathcal{M} = (D, rew)$, *kde* $D = (S, s_{init}, P)$ *odpovídá realizaci* $r \in R^{\mathcal{M}}$ *z příkladu 2.1.5, pro kterou platí* $X = 0$ *a* $Y = 0$. *Hodnoty rewardu pro jednotlivé stavy* s_x *jsou* $rew(s_x) = 1$, *pro stavy* t_x *platí* $rew(t_x) = 0$. *Rozložení pravděpodobností přechodů mezi jednotlivými stavy odpovídá popisu ve grafu 2.3a. Nechť* φ *je safety reward vlastnost* $\varphi = R_{\leq 3,8}(\Diamond T)$, *kde* $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$. *Pro vytvoření protipříkladu pro MRM* $\mathcal{M}$ *a reward vlastnost* φ *musíme nalézt takovou množinu kritických stavů* C, *aby pro podsystém* $\mathcal{M} \downarrow C$ *platilo* $\mathcal{M} \downarrow C \not\models \varphi$. *Tedy pro* $\mathcal{M} \downarrow C$ *musí platit* $ExpRew^{\mathcal{M}}[s_0 \models \Diamond(T \cup t)] \geq 3,8$. *Mějme množiny stavů* $C_0 = \{s_0, s_1, s_2, s_3, s_6\}$, $C_1 = \{s_0, s_1, s_3, \}$ *a* $C_2 = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$. *Jelikož* $ExpRew^{\mathcal{M}\downarrow C_0}[s_0 \models \Diamond(T \cup t)] = 3,\overline{3}$, *podsystém* $\mathcal{M} \downarrow C_0$ *nesplňuje podmínku* $ExpRew^{\mathcal{M}\downarrow C_0}[s_0 \models \Diamond(s_3 \cup t)] \geq 3,8$ *a tedy není protopříkladem pro MRM* $\mathcal{M}$ *a reward vlastnost* φ. *Stejně je tomu u podsystému* $\mathcal{M} \downarrow C_1$. *Pro ten platí* $ExpRew^{\mathcal{M}\downarrow C_1}[s_0 \models \Diamond(T \cup t)] = 1,\overline{9}$ *a tedy také nesplňuje specifikovanou podmínku. Jinak je tomu u podsystému* $\mathcal{M} \downarrow C_2$, *jehož očekávaná hodnota rewardů je* $3,\overline{9}$. *Z toho důvodu splňuje podmínku* $ExpRew^{\mathcal{M}\downarrow C_1}[s_0 \models \Diamond(s_3 \cup t)] \geq 3,8$ *a tedy je protipříkladem pro MRM* $\mathcal{M}$ *a reward vlastnost* φ.

3.1.2 Použití hybridní metody

V předchozím příkladu jsme našli protipříklad pro realizaci $r \in R^{\mathcal{M}}$ tvořený množinou kritických stavů $C_2 = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$. Tento protipříklad ale obsahuje všechny stavy, pro které platí $s_x \notin T$ a tím pádem je jeho tvorba pro účely pravděpodobnostní syntézy nevýhodná. Nedochází zde k žádnému zobecnění důvodu nevyhovění požadované specifikace a tím pádem by žádným způsobem nezrychlil průchod množinou prohledávaných řešení. Na chování této realizace mají vliv oba dva parametry. Tím pádem by se z množiny prohledávaných řešení odstranila pouze tato konkrétní realizace a tvorba protipříkladu by žádným způsoben nezefektivnila pravděpodobnostní syntézu. Cílem této práce je najít způsob nalezení menších a obecnějších protipříkladů.

K dosažení tohoto cíle jsem použil způsob tvorby protipříkladů inspirovaný novou metou pro pravděpodobnostní syntézu 2.6.3. Ten při tvorbě protipříkladů využívá informace o konkrétní rodině Markovových řetězců a díky tomu je vytváří přesně pro použití v dané rodině Markovových řetězců. Výše uvedené protipříklady jsou použitelné pro libovolný MRM. Pokud jsou podřetězcem daného MRM, můžeme prohlásit, že tento MRM nesplňuje specifikované požadavky. Při pravděpodobnostní syntéze ale známe rodinu Markovových řetězců se kterou pracujeme. Tím pádem máme možnost využít znalosti o dané rodině Markovových řetězců a vytvářet obecnější protipříklady.

Ukažme si to na příkladu 3.1.2. Z pozorování všech realizací $r \in R^{\mathcal{D}}$ 2.3 lze vypozorovat, že všechny realizace které mají totožné stavy s_0, s_1 a s_3 jako zkoumaná realizace, mají očekávanou hodnotu rewardu větší než $3,8$ a tedy nevyhovují specifikované vlastnosti. Výhodou takového protipříkladu by bylo, že je ovlivňován pouze parametrem X. Jelikož není ovlivňován parametrem Y, tak by při jeho použití bylo možné z množiny prohledáva-

ných řešení odebrat polovinu řešení. To znamená, že by to byl ideální protipříklad pro účely pravděpodobnostní syntézy. Problém je, že tento protipříklad sám o sobě specifikace neporušuje. Existuje ale způsob jakým ověřit, že žádná realizace $r \in R^{\mathcal{D}}$ nemůže požadované specifikace splňovat. Touto metodou se budeme zabývat v následující části.

Na tomto příkladu si můžeme všimnout ještě dalších informací, které pomohou k vytvoření menšího protipříkladu. Všimněme si, že očekávaná hodnota rewardu stavu s_4 nebo s_5 je vždy rovna jedné. Také si všimněme, že minimální získaná očekávaná hodnota rewardu stavu s_2 je $2,\overline{6}$. Tedy je zaručeno, že se po vstupu do těchto stavů v rámci rodiny $\mathcal{M}$ získají alespoň zmíněné hodnoty rewardu. Tím pádem známe minimální možnou očekávanou hodnotu rewardu pro dané stavy. Pokud bychom u protipříkladu C_1 těmto stavům přiřadily hodnoty rewardu odpovídající konkrétním předem zmíněným hodnotám, získali bychom výslednou hodnotu očekávaného rewardu $3,8095$. Tato hodnota již splňuje podmínku $3,8095 > 3,8$ a tedy porušuje specifikovanou podmínku a daný podsystém by mohl být použit jako protipříklad pro realizace $r \in R^{\mathcal{D}}$ a vlastnost λ.
K rozpoznání zda množina expandovaných stavů je dostačující k vytvoření protipříkladu v kontextu dané rodiny Markovových řetězců využívám přesměrované Markovovy reward modely. Jedná se o podobný princip jako jsou přesměrované Markovovy řetězce 2.6.4. Rozdílem je, že zde není funkce γ definující pravděpodobnost přechodu do cílového stavu. Místo ní je zde funkce ψ udávající hodnotu rewardu.

Definice 3.1.3 ***(Přesměrované MRM).*** *Nechť $\mathcal{M} = (D, rew)$ je MRM kde $D = (S, s_{init}, \mathbf{P})$ a stav $s_\top \notin S$. Mějme množinu expandovaných stavů $C \subseteq S$ a funkci $\psi : S \backslash C \to \mathbb{R}$. Přesměrováním Markovova reward modelu $\mathcal{M}$ použitím množiny stavů C a funkce ψ vzniká MRM $\mathcal{M} \downarrow C[\psi] = (D', rew^C_\psi)$ kde $D' = (S \cup \{s_\top\}, s_{init}, \mathbf{P}')$, kde pro P' platí $P'(s,t) = P(s,t)$ pokud $s \in C$ a v ostatních případech $P'(s, s_\top) = 1$. A rew^C_ψ je definován následovně:*

$$rew^C_\psi(s) = \begin{cases} rew(s) & : s \in C \\ \psi(s) & : s \in S \backslash C \\ 0 & : s = s_\top \end{cases}$$

Funkce $\psi(s)$ zde vrací příslušné hodnoty $lrb(s)$ v případě safety vlastnosti, nebo $urb(s)$ v případě liveness vlastnosti. Tyto hodnoty jsou získané pomocí zjemňování abstrakce. Hodnota $lrb(s)$ značí minimální možnou očekávanou hodnotu rewardu pro stav s, která může v libovolné realizaci dané rodiny Markovových řetězců nastat. Hodnota $lrb(s)$ naopak značí maximální možnou hodnotu očekávaného rewardu. Neboli platí $lrb^{\mathcal{R}}(s) \leq \min_{r \in \mathcal{R}} R[\mathcal{M}_r, s \models \Diamond T] \leq \max_{r \in \mathcal{R}} R[\mathcal{M}_r, s \models \Diamond T] \leq urb^{\mathcal{R}}(s)$. Stav $s_\top$ se zde přidává do množiny cílových stavů. Používá se zde upravená požadovaná vlastnost φ', která akceptuje stav $s_\top$ jako cílový stav. Přechody $s \to s_\top$ tvoří společně s rewardy určenými funkcí ψ zkratky, simulující chování daného MRM po přechodu do stavu s. V případě safety vlastnosti tyto zkratky symbolizují minimální možnou očekávanou hodnotu rewardu pro daný stav. Díky přidání této hodnoty rewardu mají tyto podsystémy vyšší hodnotu očekávaného rewardu a tím pádem i větší pravděpodobnost, že budou moct být protipříkladem pro danou podmínku. Zároveň je ale zaručeno, že nenastane situace, kdy by některá z realizací obsahující daný podsystém měla očekávanou hodnotu rewardu menší než daný protipříklad.

Tvorba protipříkladu začíná tvorbou přesměrovaného MRM pro množinu stavů $C = \emptyset$.

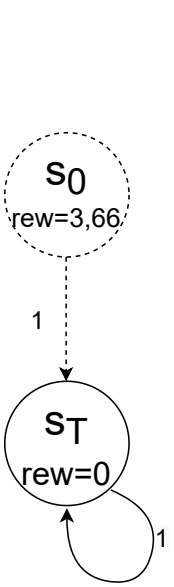
Ten je tvořen pouze stavem s_{init} a cílovým stavem t. Hodnota rewardu stavu s_{init} odpovídá hodnotě lrb nebo urb podle toho, zda se jedná o safety nebo liveness vlastnost. Pro hodnotu rewardu stavu t platí $rew(t) = 0$. Pokud tento nově vytvořený MRM nesplňuje specifikovanou vlastnost $\varphi = R_{\bowtie\lambda}(\Diamond T)$, je použit jako protipříklad v kontextu dané rodiny Markovových reward modelů. V takovém případě víme, že žádná realizace $r \in R^{\mathcal{M}}$ nesplňuje zadanou specifikaci a tedy hledané řešení neexistuje. V případě kdy $\mathcal{M} \downarrow C[\psi]$ požadovanou specifikaci splňuje je potřeba zvětšit množinu stavů, ze které se vytvoří nový přesměrovaný MRM. Přidávání nových stavů do množiny, pro kterou se vytváří přesměrované MRM, budeme nazývat expanzemi. Množiny nových stavů přidávaných při expanzi množiny C, jsou vybírány tak, aby byl výsledný MRM ovlivňován co nejmenším počtem parametrů a tím pádem byl co nejobecnější pro účely pravděpodobnostní syntézy. Pomocí této nově vzniklé množiny stavů se vytvoří nový přesměrovaný MRM. U něj kontrola splnění vlastnosti proběhne stejným způsobem. Pokud podmínku φ nesplňuje, může být použit jako protipříklad. V opačném případě dochází k další expanzi stavů. Takto se pokračuje až do stavu, kdy aktuální přesměrovaný MRM vlastnost φ nesplňuje a může být použit jako protipříklad pro účely dané pravděpodobnostní syntézy. Postup vytváření protipříkladů pro

Algoritmus 4: Tvorba protipříkladů pro rodinu MRM

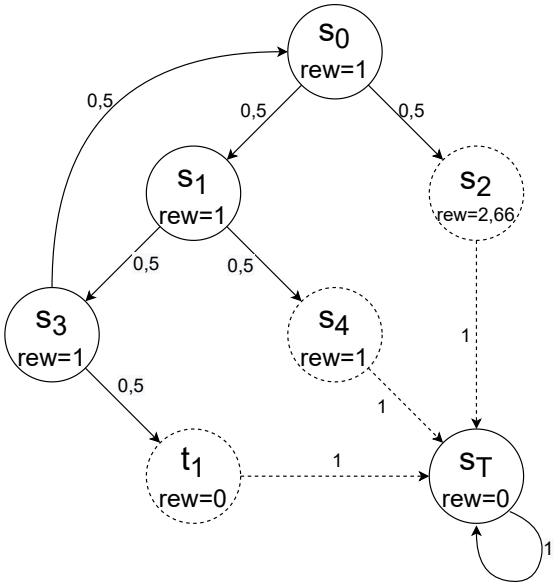
```
1  CEforMRM(Reward vlastnost φ = R_{⋈λ}(◇T),MRM M, ψ)
2  C ← ∅
3  φ' = R_{⋈λ}(◇(T ∪ t))
4  while true do
5  |  M_ψ ← M ↓ C[ψ]
6  |  if M_ψ ⊭ φ' then
7  |  |  return C
8  |  end if
9  |  C ← Expand(C, M)
10 end while
```

MRM s využitím hybridní metody je popsán algoritmem 4. Vstupem tohoto algoritmu je reward vlastnost $\varphi = R_{\bowtie\lambda}(\Diamond T)$, MRM $\mathcal{M}$ který nesplňuje vlastnost φ a funkce ψ, která vrací hodnoty lrb pro konkrétní stavy v případě kdy je φ safety vlastnost, nebo příslušné hodnoty urb pokud je φ liveness vlastnost. Na začátku algoritmu proběhne inicializace množiny stavů C a upravené vlastnosti φ'. Poté následuje cyklus, ve kterém se nejdříve vytvoří přesměrovaný MRM $\mathcal{M}_\psi$ pomocí množiny stavů C, MRM $\mathcal{M}$ a funkce ψ. Pokud MRM $\mathcal{M}_\psi$ splňuje vlastnost φ', je vrácena množina stavů C, tvořící kritický podsystém pro danou rodinu MRM a vlastnost φ. Pokud MRM $\mathcal{M}_\psi$ vlastnost φ' splňuje, tak se pomocí funkce `Expand(C,`$\mathcal{M}$`)` expanduje množina stavů a opakuje se cyklus. Všimněme si, že ukončení cyklu vytváření protipříkladu je zaručené, protože víme, že MRM $\mathcal{M}$ nesplňuje vlastnost φ. Pokud tedy nastane, že množina stavů C obsahuje všechny stavy $s \in S$, je zaručeno, aby MRM $\mathcal{M}_\psi$ vlastnost φ' nesplňoval, a tedy se cyklus přerušil.

Příklad 3.1.4 *Mějme rodinu MRM $\mathcal{M} = (\mathcal{D}, rew)$, kde $\mathcal{D}$ odpovídá rodině Markovových řetězců $\mathcal{D}$ z příkladu 2.4.3. Hodnoty rewardu pro jednotlivé stavy s_x jsou $rew(s_x) = 1$, pro stavy t_x platí $rew(t_x) = 0$. Nechť φ je safety reward vlastnost $\varphi = R_{\leq 3,8}(\Diamond T)$, kde $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$. V tomto případě zvolíme realizaci $r \in R^{\mathcal{M}}$ s volbou parametrů $X = s_0, Y = s_0$. Pro tuto realizaci platí $R[\mathcal{M}_r, \Diamond(T \cup t)] = 3,\overline{9}$ a tedy nesplňuje vlastnost*



(a) $\mathcal{M} \downarrow C_0[\psi]$



(b) $\mathcal{M} \downarrow C_1[\psi]$

φ. Naším úkolem je vytvořit takový protipříklad pro danou rodinu MRM, který je ovlivněn co nejmenším počtem parametrů.

Funkcí ψ jsou stavům přiděleny hodnoty $lrb^{\mathcal{R}} : [s_0 \to 3,\overline{6}, s_1 \to 2,\overline{6}, s_2 \to 2,\overline{6}, s_3 \to 2,\overline{3}, s_4 \to 1, s_5 \to 1, s_6 \to 2,\overline{3}]$. Pomocí množiny stavů $C_0 = \emptyset$ vytvoříme přesměrovaný MRM $\mathcal{M} \downarrow C_0[\psi]$. Ten je znázorněný grafem 3.1a. Pro MRM $\mathcal{M} \downarrow C_0[\psi]$ platí $R[\mathcal{M} \downarrow C_0[\psi], \Diamond(T \cup t)] = 3,\overline{6}$. Protože je získaná hodnota očekávaného rewardu menší než 3,8 a tedy získaný MRM splňuje specifikovanou vlastnost φ nemůže být použit jako protipříklad. Proto je potřeba expandovat množinu stavů ze kterých je tvořen přesměrovaný MRM. Pro účely tohoto příkladu jsou zvoleny stavy $C_1 = \{s_0, s_1, s_3\}$ které jsou ovlivňovány pouze jedním parametrem. Pomocí těchto stavů vytvoříme přesměrovaný MRM $\mathcal{M} \downarrow C_1[\psi]$ který je znázorněný grafem 3.1b. Pro tento MRM platí $R[\mathcal{M} \downarrow C_1[\psi], \Diamond(T \cup t)] = 3,8095 > 3,8$ a tedy lze použít jako protipříklad pro danou rodinu MRM $\mathcal{M}$ a safety vlastnost φ.

3.1.3 Protipříklady pro liveness vlastnost

Můžeme jsi všimnout, že se v části o standardních způsobech tvoření protipříkladů pro MRM 3.1.1 píše pouze o safety vlastnostech. Je to z důvodu, že tímto způsobem protipříklady pro liveness vlastnost tvořit nelze. U safety vlastností je totiž dostačující najít takový podsystém zadaného MRM, který sám o sobě svojí očekávanou hodnotou rewardu převyšuje požadovanou hranici, definovanou danou safety vlastností. U liveness vlastností avšak hledáme důkaz, že MRM obsahující daný protipříklad má vždy očekávanou hodnotu rewardu menší než hraniční hodnota definovaná liveness vlastností. Očekávaná hodnota rewardu podsystému MRM je ale vždy menší nebo rovna očekávané hodnotě rewardu původního MRM. Tím pádem by nedávalo smysl vytvořit podsystém, který odporuje liveness podmínce, protože přidáním dalších stavů, je možné liveness vlastnost splnit.

Další otázkou je, zda pro tvorbu protipříkladů MRM pro liveness vlastnosti nebude problémem fakt, že pokud je pravděpodobnost dosažení cílového stavu menší než 1, je očekávaná hodnota rewardu daného MRM rovna ∞.

Při použití výše popsané metody pro tvorbu protipříkladů pro MRM pomocí informací získaných zjemňováním abstrakce 3.1.2 je ale tvorba pro liveness vlastnosti možná. V takovém případě totiž při tvorbě zkratek v přesměrovaném MRM používáme hodnoty rewardu *urb*. Ty obsahují maximální možnou očekávanou hodnotu rewardu. Tím pádem expandováním množiny stavů, ze kterých se tvoří přesměrovaný MRM, očekávaná hodnota rewardu vytvořeného MRM klesá. Je to tak z důvodu, že se hodnota očekávaného rewardu postupným expandováním stavů z původní maximální hodnoty pro všechny realizace přibližuje hodnotě daného MRM. Tím pádem odpadá výše zmíněný problém protipříkladů MRM pro liveness vlastnosti, tvořených standardním způsobem.

Tvorba protipříkladu MRM pro liveness vlastnosti je již obecně popsána v části 3.1.2. Prvním krokem pro tvorbu takového protipříkladu je vytvoření přesměrovaného MRM dané rodiny MRM pro prázdnou množinu stavů. Pokud je hodnota očekávaného rewardu vytvořeného přesměrovaného MRM menší než limitní hodnota liveness vlastnosti, tak víme že není možné aby existovala realizace dané rodiny MRM, která splňuje požadovanou podmínku. V podstatě tak vytvoříme protipříklad pro danou rodinu MRM a liveness vlastnost, který ale není závislý na žádném parametru. Pokud vytvořený přesměrovaný MRM požadovanou vlastnost splňuje, pokračuje se dál expandováním množiny stavů ze kterých se vytváří přesměrovaný MRM. Průběhem expandování dané množiny stavů se vytvořený přesměrovaný MRM stále více podobá původnímu MRM, ze kterého se daný přesměrovaný MRM tvoří. Tím pádem se postupně snižuje očekávaná hodnota rewardu. Zároveň se ale zvyšuje i počet parametrů, jejichž volba má vliv na chování daného přesměrovaného MRM. V nejhorším případě může nastat, že expanze množiny stavů pokračují až do takové situace, kdy obsahuje všechny stavy dané realizace. V takovém případě je zaručeno, že právě vytvořený MRM odporuje požadované vlastnosti, a tak je z něj možné vytvořit protipříklad. V takovém případě je ale vzniklý protipříklad ovlivňován všemi existujícími parametry, a tak z množiny prohledávaných řešení vylučuje pouze jedno řešení.

Příklad 3.1.5 *Uvažujme stejnou rodinu MRM jako v příkladu 3.1.2. Tentokrát chceme vytvořit protipříklad pro liveness reward vlastnost* $\varphi = R_{\geq 3,85}(\Diamond T)$*, kde* $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$*. V tomto případě zvolíme realizaci* $r \in R^{\mathcal{M}}$ *s volbou parametrů* $X = s_2, Y = s_1$*. Pro tuto realizaci platí* $R[\mathcal{M}_r, \Diamond(T \cup t)] = 3, \overline{6}$ *a tedy vlastnost* φ *nesplňuje. Stejně jako v předchozím příkladu je zde naším úkolem vytvořit takový protipříklad pro danou rodinu MRM, který je ovlivněn co nejmenším počtem parametrů. V tomto případě se ale nejedná o safety vlastnost, ale o liveness vlastnost.*

Funkcí ψ *jsou stavům přiděleny hodnoty rewardu* $urb^{\mathcal{R}} : [s_0 \to 3, \overline{9}, s_1 \to 2, 9524, s_2 \to 2, 9524, s_3 \to 2, \overline{9}, s_4 \to 1, s_5 \to 1, s_6 \to 2, \overline{9}]$*. Pomocí množiny stavů* $C_0 = \emptyset$ *vytvoříme přesměrovaný MRM* $\mathcal{M} \downarrow C_0[\psi]$*. Hodnota očekávaného rewardu vytvořeného přesměrovaného MRM se rovná* $R[\mathcal{M} \downarrow C_0[\psi], \Diamond(T \cup t)] = 3, \overline{9}$*. Tento MRM nelze použít jako protipříklad, protože jeho očekávaná hodnota rewardu je větší než limitní hodnota specifikované liveness vlastnosti. Tím pádem tuto vlastnost splňuje. Proto je potřeba expandovat množinu stavů a vytvořit nový přesměrovaný MRM. Pro účely tohoto příkladu jsou zvoleny stavy* $C_1 = \{s_0, s_2, s_5\}$ *které jsou ovlivňovány pouze jedním parametrem. Pomocí těchto stavů vytvoříme přesměrovaný MRM* $\mathcal{M} \downarrow C_1[\psi]$*. Pro tento MRM platí* $R[\mathcal{M} \downarrow C_1[\psi], \Diamond(T \cup t)] = 3, 8095 < 3, 85$ *a tedy lze použít jako protipříklad pro danou rodinu MRM* $\mathcal{M}$ *a liveness vlastnost* φ*.*

Dále je potřeba zjistit, jak tento způsob tvorby protipříkladů funguje v případech, kdy pro některé realizace platí $P(s_0 \models \Diamond T) < 1$. V takovém případě je jejich očekávaná hodnota

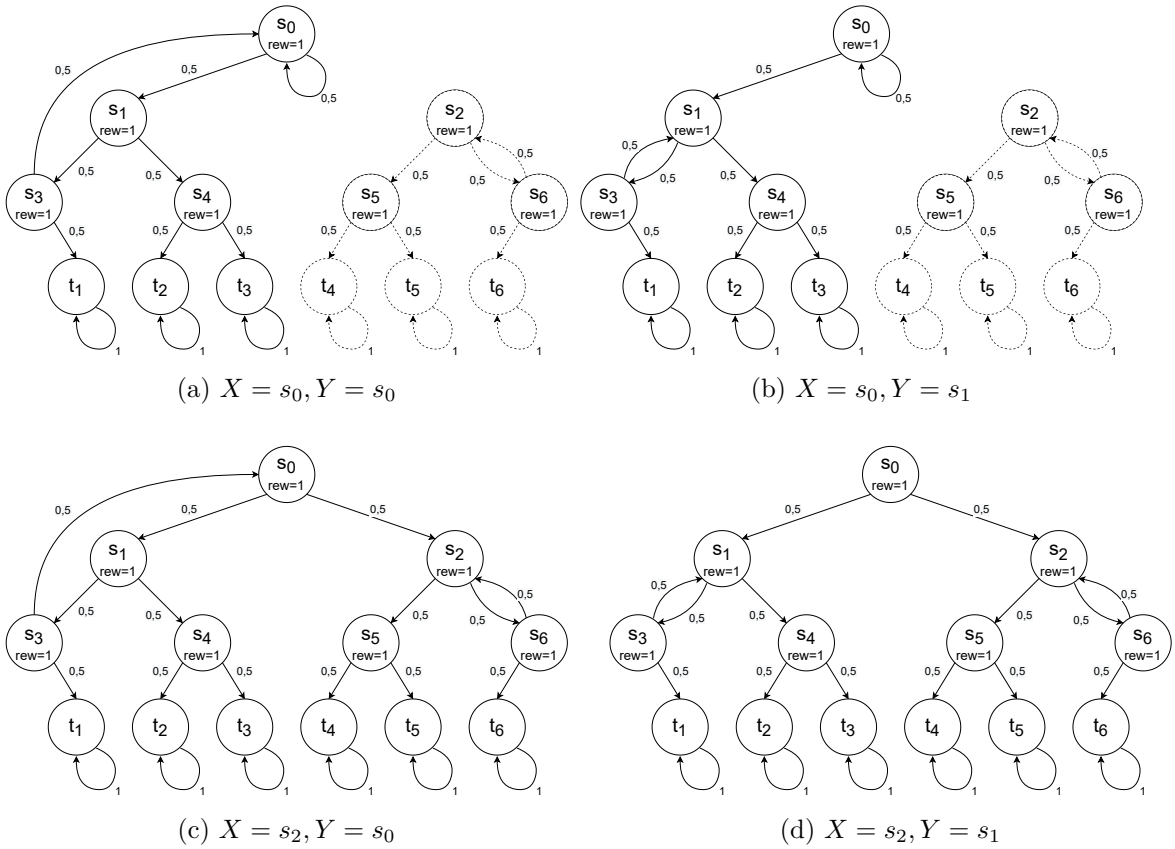
rewardu rovna nekonečnu. Na první pohled tento fakt může působit problémy. Po pozornějším prozkoumání a experimentálním vyhodnocení, jsem zjistil, že taková situace nemá na tvorbu protipříkladů příliš veliký vliv. Ve výsledku se části modelu s očekávanou hodnotou rewardu rovnající se nekonečnu chovají jako stavy s extrémně velikou hodnotou rewardu. Jediný problém může působit to, že stačí i extrémně malá pravděpodobnost přechodu do stavu s nekonečně velikou očekávanou hodnotou rewardu a výsledná očekávaná hodnota rewardu celého MRM je rovna nekonečnu. To může občas způsobit potřebu expandovat více stavů, aby se existence takového stavu odstranila. Tím pádem tato metoda může být v případě kdy pro některé realizace platí $P(s_0 \models \Diamond T) < 1$ mírně náročnější. V porovnání s metodou one-by-one je však stále extrémně efektivnější.

Příklad 3.1.6 *Mějme rodinu MRM $\mathcal{M} = (\mathcal{D}, rew)$, kde $\mathcal{D} = (S, s_{init}, K, \mathcal{B})$. S, s_{init} a K odpovídají S, s_{init} a K z příkladu 2.4.3 a funkce pro rozdělení pravděpodobnosti přechodů $\mathcal{B}$ vypadá následovně:*

$$\begin{aligned}
\mathcal{B}(s_0) &= 0,5 : k_{s_1} + 0,5 : X,\\
\mathcal{B}(s_1) &= 0,5 : k_{s_3} + 0,5 : k_{s_4},\\
\mathcal{B}(s_2) &= 0,5 : k_{s_5} + 0,5 : k_{s_6},\\
\mathcal{B}(s_3) &= 0,5 : Y + 0,5 : k_{t_1},\\
\mathcal{B}(s_4) &= 0,5 : k_{t_2} + 0,5 : k_{t_3},\\
\mathcal{B}(s_5) &= 0,5 : k_{t_4} + 0,5 : k_{s_5},\\
\mathcal{B}(s_6) &= 0,5 : k_{t_2} + 0,5 : k_{t_6},\\
\mathcal{B}(t_n) &= 1 : k_{t_n}.
\end{aligned}$$

Všechny 4 realizace dané rodiny MRM jsou znázorněny ve grafech 3.2. Hodnoty rewardu pro jednotlivé stavy s_x jsou $rew(s_x) = 1$, pro stavy t_x platí $rew(t_x) = 0$. Cílem je vytvořit protipříklad pro vlastnost $\varphi = R_{\geq 5,0}(\Diamond T)$, kde $T = \{t_1, t_2, t_3\}$. Všimněme si, že zde existují 3 absorbující stavy t_4, t_5 a t_6, které ale nejsou součástí množiny cílových stavů. Tím pádem je jejich očekávaná hodnota rewardu rovna ∞. Pro tvorbu protipříkladu si zvolíme realizaci $r \in R^{\mathcal{M}}$ s volbou parametrů $X = s_0, Y = s_1$ 3.2b. Pro tuto realizaci platí $R[\mathcal{M}_r, \Diamond(T \cup t)] = 4,\overline{6}$ a tedy vlastnost φ nesplňuje.

V tomto případě jsou stavům přiděleny hodnoty rewardu $urb^{\mathcal{R}} : [s_0 \to \infty, s_1 \to \infty, s_2 \to \infty, s_3 \to \infty, s_4 \to 1, s_5 \to \infty, s_6 \to \infty]$. Stejně jako v předchozích příkladech nejdříve vytvoříme přesměrovaný MRM pomocí množiny stavů $C_0 = \emptyset$. Ten je znázorněn grafem 3.3a. Očekávaná hodnota rewardu právě vytvořeného MRM je $R[\mathcal{M} \downarrow C_0[\psi], \Diamond(T \cup t)] = \infty$ a tedy splňuje specifikovanou vlastnost. Provedeme tedy expanzi stavů na množinu $C_1 = \{s_0\}$. Pomocí této množiny stavů vytvoříme další přesměrovaný MRM $\mathcal{M} \downarrow C_1[\psi]$ 3.3b, pro který znova platí $R[\mathcal{M} \downarrow C_1[\psi], \Diamond(T \cup t)] = \infty$ a tedy opět splňuje specifikovanou vlastnost. Je tedy potřeba znova expandovat množinu stavů, ze kterých je tvořen přesměrovaný MRM. Tentokrát je zvolena množina stavů $C_2 = \{s_0, s_1, s_3\}$. Pomocí množiny stavů C_2 je vytvořen přesměrovaný MRM $\mathcal{M} \downarrow C_2[\psi]$ 3.3c, pro který platí $R[\mathcal{M} \downarrow C_2[\psi], \Diamond(T \cup t)] = 4,\overline{6}$ a tedy nesplňuje zadanou vlastnost φ. Podsystém $R[\mathcal{M} \downarrow C_2[\psi]$ je tedy možné použít jako protipříklad pro danou rodinu MRM $\mathcal{M}$ a liveness vlastnost φ



(a) $X = s_0, Y = s_0$ (b) $X = s_0, Y = s_1$

(c) $X = s_2, Y = s_0$ (d) $X = s_2, Y = s_1$

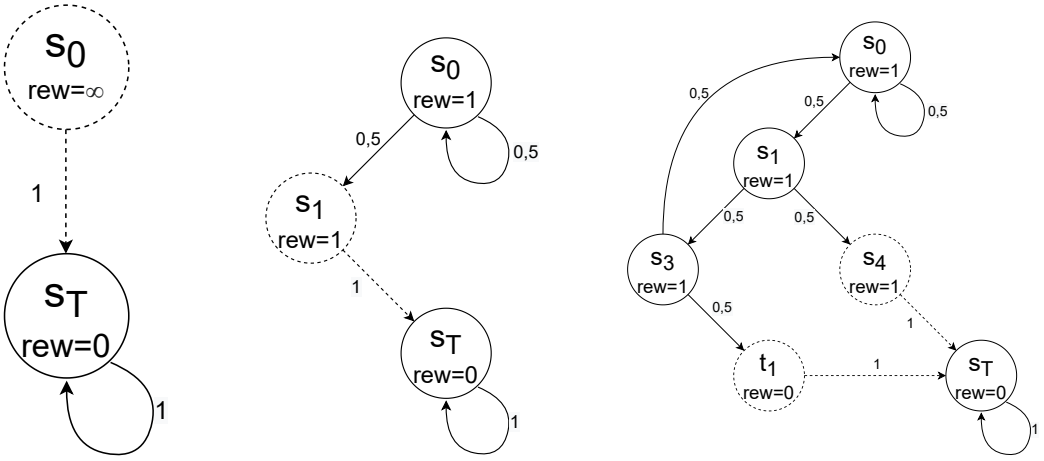
Obrázek 3.2: Realizace rodiny MRM použité pro ukázku pravděpodobnostní syntézy pro liveness reward vlastnost.

3.2 Until vlastnost

Druhým tématem, kterým jsem se zabýval v rámci této práce je rozšíření specifikačního jazyka výše zmíněné hybridní metody pro pravděpodobnostní syntézu o možnost použití until vlastností. Until vlastnosti se skládají ze dvou podvlastností a jsou platné pokud druhá podvlastnost platí aspoň v některém stavu a první podvlastnost platí ve všech předchozích stavech. To umožňuje používat komplexnější specifikaci hledaného řešení pravděpodobnostní syntézy. Toto rozšíření je klíčové, jelikož se na ověřování until vlastností redukuje libovolné ověřování PCTL vlastnosti.

Po zkoumání způsobů řešení tohoto problému jsem došel k závěru, že na řešení tohoto problému není třeba existující metodu více upravovat. Hybridní metoda pro tento účel funguje dobře, a tak není potřeba vymýšlet novu metodu. Ve výsledku jsem se u tohoto úkolu zabýval spíše implementací tohoto rozšíření do existujícího nástroje pro pravděpodobnostní syntézu.

Tvorba protipříkladů pro until vlastnosti je popsána algoritmem 5. Na vstupu je pravděpodobnostní until vlastnost $\varphi = P_{\bowtie\lambda}(\Psi \mathbf{U} \Phi)$, DTMC D a funkce γ. DTMC D musí být obsaženo v množině možných řešení prohledávané danou pravděpodobnostní syntézou. Také DTMC D nesmí splňovat zadanou vlastnost φ. Funkce γ pro konkrétní stavy vrací jejich hodnoty *lub* pokud je φ safety vlastnost nebo *uub* se jedná o liveness vlastnost. Hodnota *lub* obsahuje minimální možnou pravděpodobnost splnění until vlastnosti φ z daného stavu



(a) $\mathcal{M} \downarrow C_2[\psi]$ (b) $\mathcal{M} \downarrow C_1[\psi]$ (c) $\mathcal{M} \downarrow C_0[\psi]$

Algoritmus 5: Tvorba protipříkladů pro until vlastnost

```
1 CEforUntil(Prvděpodobnostní until vlastnost φ = P_{⋈λ}(Ψ U Φ),DTMC D, γ)
2 C ← ∅
3 while true do
4 |   D_γ ← D ↓ C[γ]
5 |   if D_γ ⊭ φ' then
6 |   |   return C
7 |   end if
8 |   C ← Expand(C, D)
9 end while
```

získanou pomocí zjemňování abstrakce. Hodnota *uub* naopak obsahuje pravděpodobnost maximální.

Na začátku se vytvoří prázdná množina stavů C_0. Pro tuto množinu stavů se vytvoří přesměrovaný DTMC $D \downarrow C[\gamma]$. Pokud právě vytvořený DTMC $D \downarrow C[\gamma]$ nesplňuje vlastnost φ, je vhodným protipříkladem pro danou rodinu DTMC a zadanou vlastnost φ. V takovém případě je výsledkem protipříklad $D \downarrow C[\gamma]$ a algoritmus končí. Pokud však until vlastnost φ splňuje, nemůže být použit jako protipříklad. Je tedy nutné expandovat množinu stavů C a provést u ní obdobnou kontrolu, zda splňuje vlastnost φ. Tento postup se opakuje až do doby, kdy vytvořený přesměrovaný DTMC $D \downarrow C[\gamma]$ nesplňuje vlastnost φ. V ten moment je výsledkem algoritmu právě vytvořený DTMC $D \downarrow C[\gamma]$ a algoritmus je ukončen. Ukázku tvorby protipříkladu pro pravděpodobnostní until vlastnost si ukážeme v následujícím příkladu.

Příklad 3.2.1 *Mějme stejnou rodinu DTMC jako v příkladu 2.4.3. Chceme vytvořit protipříklad pro safety until vlastnost* $\varphi = P_{\leq 4,9}(\neg s_2 \ \mathbf{U} \ T)$*, kde* $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$*. V tomto případě zvolíme realizaci* $r \in R^{\mathcal{D}}$ *s volbou parametrů* $X = s_2, Y = s_1$*. Realizace r vlastnost* φ *nesplňuje, jelikož pro tuto realizaci platí* $P[\mathcal{D}_r, (\neg t_2 \ \mathbf{U} \ (T \cup t))] = 5,0$*. Úkolem je vytvořit takový protipříklad pro danou rodinu DTMC, který je ovlivněn co nejmenším počtem parametrů.*

Funkcí γ *jsou tentokrát stavům přiděleny hodnoty rewardu* $lub^{\mathcal{R}} : [s_0 \to 0,4285, s_1 \to 0,8571, s_2 \to 0, s_3 \to 0,7142, s_4 \to 1, s_5 \to 1, s_6 \to 0,7142]$*. Pomocí prázdné množiny stavů*

$C_0 = \emptyset$ vytvoříme přesměrovaný DTMC $D \downarrow C_0[\gamma]$. Hodnota očekávaného rewardu vytvořeného přesměrovaného DTMC se rovná $R[D \downarrow C_0[\gamma], (\neg t_2 \ \mathbf{U} \ (T \cup t))] = 0,4285$. Vzhledem k tomu, že je tato hodnota menší než limit pro splnění safety until vlastnosti, tento vytvořený DTMC specifikovanou vlastnost splňuje a tedy nemůže být použit jako protipříklad pro danou situaci. Proto je potřeba expandovat množinu stavů a pomocí ní vytvořit nový přesměrovaný DTMC. Pro účely tohoto příkladu jsou zvoleny stavy $C_1 = \{s_0, s_1, s_3\}$. Pomocí nově expandované množiny stavů vytvoříme přesměrovaný DTMC $D \downarrow C_1[\gamma]$. Pro tento DTMC platí $R[D \downarrow C_1[\gamma], (\neg t_2 \ \mathbf{U} \ (T \cup t))] = 0.4285714$. Tím pádem tento DTMC nesplňuje zadanou vlastnost φ a tedy může být použitý jako protipříklad pro danou rodinu DTMC $\mathcal{D}$ a until vlastnost φ.

3.3 Implementace

Cílem implementace bylo rozšíření specifikačního jazyka nástroje pro syntézu pravděpodobnostních modelů PAYNT[1]. Tento nástroj umožňuje provádět pravděpodobnostní syntézu pomocí metody kombinující induktivní a deduktivní přístup, popsanou v sekci 2.6.3. Tento nástroj ke tvorbě protipříkladů využívá program Storm [5], který je napsaný v programovacím jazyce C++. Náplní této práce byla implementace rozšíření části programu Storm která vytváří protipříklady. Díky implementaci klíčových algoritmů pro tvorbu protipříkladů je nástroj schopný prováděl pravděpodobnostní syntézu pomocí hybridní metody pro reward i until vlastnosti.

Pro rozšíření o možnost použití rewardů bylo potřeba implementovat změny v části týkající se tvorby protipříkladů. Implementoval jsem tvorbu přesměrovaných MRM a jejich přetváření při expandování stavů, ze kterých jsou tvořeny.

U imlementace rozšíření o možnost použití until vlastností tyto změny nebyli potřeba, protože se pro jejich zpracování dá použít stejné metody jako obvykle. Zato zde bylo potřeba implementovat zpracování until požadavků a označení stavů zpracovávaného DTMC, tak aby bylo možné kontrolovat obě podvlastnosti dané until vlastnosti.

[1] `https://github.com/gargantophob/synthesis`

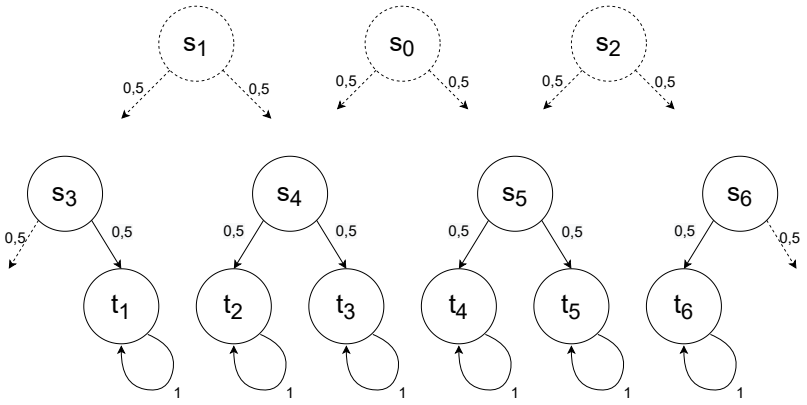
Kapitola 4

Experimentální vyhodnocení

Tato kapitola pojednává o experimentech ověřujících efektivitu implementovaného rozšíření. V této kapitole je popsáno několik experimentů ověřujících efektivitu navržených metod pro pravděpodobnostní syntézu umožňující práci s rewardy a until vlastnostmi. V první části se zaměřím na experimenty využívající rewardy. Poté následují experimenty s využitím until vlastností.

Experimenty probíhaly na počítači s procesorem Intel Core i7-8550U s operačním systémem Linux Ubuntu 20.04. K experimentům jsem použil 3 pravděpodobnostní modely `dice`, `herman` a `linka`. Následující část textu tyto modely popisuje.

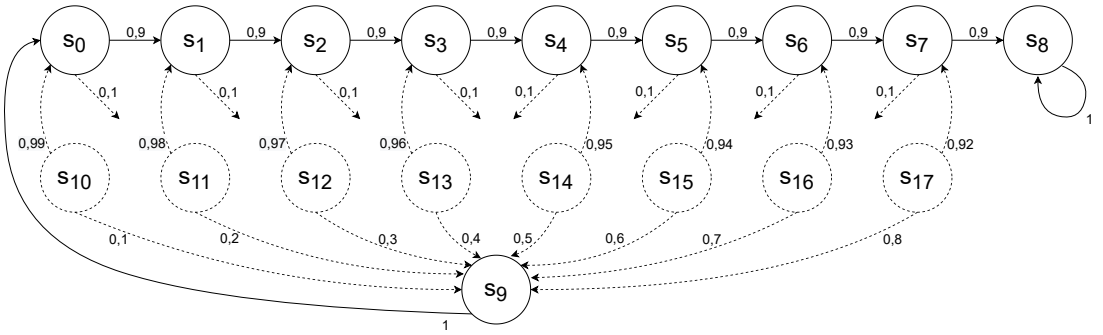
Prvním modelem využitým k experimentálnímu vyhodnocení je model `dice`, simulující hod kostkou pomocí série hodu mincí. Tento model se skládá celkem ze dvanácti stavů. Stavy tohoto modelu se dělí do 3 skupin. První skupinu tvoří 6 absorpčních stavů symbolizujících výsledek hodu kostkou. Tyto stavy mají předem definované chování, které není ovlivněno volbou parametrů. Dalším skupinu tvoří 4 stavy ze kterých vedou přechody do absorpčních stavů. Tyto přechody jsou předem definovány. Přechody do ostatních stavů jsou ale závislé na volbě parametrů. Poslední skupinu tvoří 3 stavy, které mají všechny přechody závislé na volbě parametrů, a tedy je jejich chování přímo ovlivněné výsledkem syntézy. Všechny stavy které nejsou absorpční mají právě 2 přechody s pravděpodobností 0,5. Tím pádem symbolizují hod kostkou. Strukturu tohoto modelu popisuje obrázek 4.1.



Obrázek 4.1: Model `dice`. Přerušované šipky značí přechod přímo ovlivněný volbou parametru. Počátečním stavem je stav s_0

Model `dice` má celkem 8 parametrů. Každý parametr má celkem 7 možných hodnot, které může nabývat. Existuje tedy 7^8 realizací dané rodiny MRM, vytvořené pro pravděpodobnostní syntézu tohoto modelu. Při syntéze je tedy nutné prohledat 5764801 možných řešení. Průměrná velikost DTMC u rodiny DTMC daného modelu je 9.

Pro účely demonstrace syntézy s využitím until vlastností jsem vytvořil nový model `linka` znázorněný grafem 4.2, který modeluje proces výroby produktu na výrobní lince. Pro výrobu produktu je potřeba podstoupit 7 fází výroby. Každá z těchto fázi je zastoupena jedním stavem. Tyto stavy mají určitou pravděpodobnost přechodu do následující fáze výroby, ale také je zde definována pravděpodobnost naskytnutí chyby a potřeby produkt opravit. Možnosti opravy produktu jsou modelovány další množinou stavů. Každý ze stavů symbolizujících opravu produktu má stanovené pravděpodobnosti pro zdaření opravy a pro neopravitelné poškození produktu. Tyto pravděpodobnosti se u každého ze stavů liší. Také se u každého z těchto stavů liší, do které fáze výroby produktu se přechází v případě úspěšné opravy produktu. V případě neopravitelné chyby se přechází zpět do první fáze výroby produktu. Cílem pravděpodobnostní syntézy tohoto modelu je nastavit konkrétní stav opravy produktu pro jednotlivé stavy fáze výroby. Jelikož stavy opravy s přechodem do počátečních fází výroby mají menší pravděpodobnost chyby než stavy opravy ze kterých se při úspěšné opravě přechází do pozdějších fází výroby, tak je pro různé fáze výroby výhodné použít různé stavy opravy. Tento model obsahuje 7 parametrů, 40320 možných řešení a průměrná velikost výsledného DTMC je 12 stavů.



Obrázek 4.2: Model `linka`. Stavy $s_0 - s_7$ zde značí jednotlivé fáze výroby produktu, stavy $s_{10} - s_{17}$ značí konkrétní způsoby opravy produktu, stav s_9 značí zničení produktu a stav s_8 značí úspěšné ukončení výroby produktu. Přerušované šipky které nevedou do žádného stavu značí přechody mezi stavy ovlivněné volbou parametrů. Počátečním stav je s_0.

Dalším modelem využitým pro experimentální vyhodnocení je model `herman`. Ten modeluje Hermanův algoritmus [10] využívající se pro stabilizaci kruhu procesů. Ten je stabilizovaný pokud pouze jeden z procesů daného kruhu vlastní token. Tento token by měl být předáván všem procesům v kruhu férovým způsobem. Bylo dokázáno, že tento model představuje těžký syntetizační problém, a tedy dovoluje ověřit efektivitu navržených rozšíření. Při experimentech používám 4 verze tohoto modelu s různě velikým počtem členů rodiny MRM, a tedy i různou obtížností syntézy tohoto modelu. Nejmenší modelem je `herman1`, který má celkem 500 realizací. Dále jsem použil model `herman2` s 80000 realizacemi a `herman3` který má realizací 3125000. Největším modelem je `herman4` který má 6480000 realizací. Průměrná velikost realizací je pro všechny tyto modely přibližně 1000 stavů.

4.1 Experimenty s reward vlastnostmi

Nejprve zde popíši experimenty na malém modelu `dice`. Prvním experimentem, který zde předvedu, je použití implementovaného nástroje k syntéze tohoto modelu, tak aby výsledné řešení simulovalo spravedlivý hod šestistěnnou kostkou. Také je po tomto modelu požadováno, aby jeho průměrný počet kroků nepřesahoval hodnotu 3,7. Tento experiment ukazuje, zda správně funguje použití safety reward vlastností při použití hybridní metody. Pro získání realizace rodiny MRM splňující dané požadavky je potřeba správně specifikovat vlastnosti, které musí splňovat. Nejdříve jsem specifikoval požadované chování pomocí šesti pravděpodobnostních vlastností, které určují, že výsledná pravděpodobnost přechodu do každého z absorpčních stavů je větší než 0,165. Dále jsem specifikoval reward vlastnost, která povoluje pouze výsledné MRM mající očekávanou hodnotu rewardu menší než 3,7. Jelikož zde každý neabsorpční stav má nastavenou hodnotu rewardu na 1, tak tato reward vlastnost omezí průměrný počet kroků hledaného řešení na 3,7.

Časy běhu různých syntetizačních metod jsou shrnuty v tabulce 4.1 pod názvem **dice: reward safety**. Při srovnání těchto časů si můžeme všimnout rozdílu časů feasibility syntézy a optimální syntézy při použití hybridní metody. Tento rozdíl je způsoben tím, že optimální syntéza pokračuje i po nalezení první realizace splňující požadované specifikace. Optimální syntéza pokračuje až do případu, kdy jsou ověřené nebo pomocí protipříkladů vyloučené všechna možná řešení a je možné vrátit optimální řešení tohoto problému. Také si můžeme všimnout, že je zde pouze jeden čas pro použití metody one-by-one. Tento čas značí potřebnou dobu pro kontrolu všech realizací a tedy přesně odpovídá době potřebné pro optimální syntézu daného problému za použití metody one-by-one. Feasibility syntéza by v tomto případě tedy mohla trvat maximálně takto dlouhou dobu. Specifikované řešení by ale mohla najít i dříve a to pouze v závislosti na pořadí kontrolovaných řešení. Z těchto měření tedy vychází, že při použití hybridní metody je optimální syntéza až 22krát efektivnější než při použití metody one-by-one.

Druhým experimentem ověřuji efektivitu hybridní metody při využití liveness reward vlastnosti. Zde je také cílem pomocí modelu `dice` simulovat hod spravedlivou šestistěnnou kostkou. Tentokrát ale požadujeme, aby byl průměrný počet kroků větší než 5. Pro specifikaci této podmínky použijeme liveness reward vlastnost. Časy běhu různých syntetizačních metod jsou opět shrnuty v tabulce 4.1, tentokrát pod názvem **dice: reward liveness 1**. V porovnání s předchozím experimentem jsou tyto časy mírně delší, ale řádově jsou na tom stejně. Můžeme si všimnout, že čas při použití metody one-by-one pro optimální syntézu je totožný s časem v předchozím experimentu. To je dáno tím, že se jedná o stejný model a tedy kontrola všech realizací metodou one-by-one probíhá obdobným způsobem. V tomto případě je použití hybridní metody pro optimální syntézu 13krát efektivnější.

Další experiment slouží k ověření efektivity tohoto nástroje při použití liveness reward vlastnosti v případě, kdy pro některé realizace platí, že některé z absorpčních stavů nejsou součástí množiny cílových stavů. V případě kdy v MRM nastane taková situace, je očekávaná hodnota rewardu takového MRM rovna ∞. Tento experiment dokazuje, že tento fakt při použití mého rozšíření hybridní metody nevytváří podstatné problémy.

Opět je zde použit model `dice`. Tentokrát se syntézou hledá řešení simulující hod trojstrannou kostkou. Do množiny cílových stavů tentokrát patří pouze 3 absorpční stavy, které symbolizují výsledky hodu kostkou. Pro syntézu byly použity pravděpodobnostní vlastnosti požadující pravděpodobnost přechodu do každého ze tří cílových stavů větší než 0,33. Také

	one-by-one opt.	hybridní m.	hybridní m. opt.
dice: reward safety	34h	255s	93min
dice: reward liveness 1	34h	678s	155min
dice: reward liveness 2	34h	50s	32min
dice: until	34h	0,52s	12,43s
herman1	285s	18s	15s
herman2	78min	90s	100s
herman3	51h	141s	36min
herman4	106h	183s	4h
linka: safety	15min	0,36s	0,16s
linka: liveness	16min	0,33s	0,09s

Tabulka 4.1: Výsledky experimentů. Ve sloupci **one-by-one opt.** jsou doby trvání optimální syntézy daných příkladů pomocí metody one-by-one. Ve sloupci **hybridní m.** jsou doby trvání syntézy pomocí hybridní metody a ve sloupci **hybridní m. opt.** jsou časy při použití hybridní metody pro optimální syntézu.

je zde definována liveness reward vlastnost požadující očekávanou hodnotu rewardu větší než 7.

Z výsledků v tabulce 4.1 vyplývá, že časy při použití hybridní metody jsou v tomto případě kratší než v předchozích dvou experimentech i když je jedná o experimenty nad stejným modelem. Takto rychlé nalezení odpovídajícího řešení lze odůvodnit tím, že existuje veliké množství realizací dané rodiny MRM splňující specifikované vlastnosti. Při použití metody one-by-one pro optimální syntézu trvala kontrola všech řešení opět stejně dlouhou dobu. V tomto případě bylo nalezení optimálního řešení pomocí hybridní metody přibližně 64krát efektivnější.

Dále následuje experiment testující efektivitu hybridní metody při kombinaci použití until i reward vlastností. Zde je opět použit model `dice` na modelování šestistěnné kostky s maximální očekávanou hodnotou rewardu 3,7. Tentokrát jsou ale specifikace požadovaných pravděpodobností přechodů do cílových stavů omezeny pomocí použití until vlastností. Je zde specifikováno, že pro přechod do stavů t_1, t_2 a t_3 nesmí být použit stav s_2. Obdobně pro přechod do stavů t_4, t_5 a t_6 se nesmí použít stav s_1.

Časy běhu různých syntetizačních metod jsou shrnuty v tabulce 4.1 pod názvem **dice: until**. Můžeme si všimnout, že jsou v tomto případě časy výrazně menší i když se opět jedná o experiment se stejným modelem. Toto extrémní zrychlení je z důvodu, že zadané until vlastnosti přesněji specifikují požadavky a tak jsou při hybridní metodě tvořeny konkrétnější protipříklady. V tomto případě je použití hybridní metody pro optimální syntézu přibližně 1000krát efektivnější než použití metody one-by-one.

Dále jsem realizoval experimenty na modelu `herman`. Cílem tohoto experimentu je pomocí pravděpodobnostní syntézy nalézt takovou realizaci zadané rodiny MRM, která po uběhnutí jednoho kola tvoří stabilizovaný kruh a splňuje požadované specifikace ohledně očekávané hodnoty rewardu. Tento experiment jsem realizoval na modelech s různou velikostí prohledávané rodiny MRM. Časy běhu různých syntetizačních metod jsou opět shrnuty v tabulce 4.1, tentokrát pod názvy **herman1**, **herman2**, **herman3** a **herman4**. Z těchto výsledků je patrné, že je použití hybridní metody efektivnější než metoda one-by-one i při

práci se složitějším modelem. Nejmenší rozdíl efektivity meody one-by-one a hybridní metody u optimální syntézy je u modelu **herman1**, kde je hybridní metoda 19krát efektivnější než metoda one-by-one. Naopak největší rozdíl je u modelu **herman3**, kde je použití hybridní metody efektivnější 85krát. Můžeme si všimnout, že rozdíl časů běhu feasibility a optimální syntézy roste s přibývajícím počtem možných realizací daného modelu.

λ	one-by-one	hybridní m.
8,4	78min	93s
8,2	78min	84s
8,0	78min	70s
7,5	78min	26s
7,0	78min	26s

Tabulka 4.2: Výsledky experimentů s modelem **herman2**.

Na modelu **herman2** jsem dále provedl řadu experimentů, které slouží k ověření efektivity rozšíření hybridní metody v případech, kdy je specifikována taková safety reward vlastnost kterou nesplňuje žádné řešení ze zadané rodiny. Shrnutí výsledků těchto experimentů je v tabulce 4.2. Při použití metody one-by-one je doba trvání u všech běhů stejná. To je zapříčiněno tím, že je ve všech případech potřeba postupně kontrolovat celou množinu řešení. Při použití hybridní metody se časy běhů při použití různých hodnot λ liší. V takovém případě při použití hodnoty λ blízké očekávané hodnotě rewardu optimálního řešení trvala syntéza přibližně stejně dlouhou dobu jako při optimální syntéze. Snižováním hodnoty λ se časy syntézy postupně zmenšují.

4.2 Experimenty s until vlastnostmi

Typickým využitím until vlastností je specifikování požadované pravděpodobnosti pro dosažení cílového stavu bez přechodu do chybového stavu. Pro účely demonstrace takovéto situace jsem vytvořil model `linka`. V experimentu pracujícím s modelem `linka` je úkolem pravděpodobnostní syntézy najít takovou kombinaci zvolených postupů opravy pro jednotlivé fáze výroby, tak aby pravděpodobnost dokončení výroby bez zničení produktu byla co nejmenší.

Nejdříve jsem provedl pravděpodobnostní syntézu pro model `linka`, kde jsem hledal řešení s pravděpodobností dokončení výroby bez zničení produktu větší než 0,9. Poté jsem provedl optimální syntézu metodou one-by-one a hybridní metodou. Časy běhu těchto experimentů jsou v tabulce 4.1 pod názvem **linka: safety**. Z těchto výsledků vyplývá, že je v tomto případě použití hybridní metody pro optimální syntézu až 2 500krát rychlejší než použití metody one-by-one.

Obdobný experiment jsem provedl i pro ověření efektivity při použití liveness until vlastnosti. Tentokrát pravděpodobnostní syntéza hledala řešení s pravděpodobností dokončení výroby produktu bez jeho zničení menší než 0,7. V tomto případě bylo použití hybridní metody pro optimální syntézu až 10 000krát efektivnější než použití metody one-by-one.

Kapitola 5

Závěr

Tato práce se zabývá syntézou pravděpodobnostních programů s optimální cenou. Nejdříve jsem nastudoval existující metody pro syntézu pravděpodobnostních programů s důrazem na metody založené na MDP abstrakci a protipříkladech. Po vyhodnocení nedostatků těchto metod jsem se zaměřil na metodu pro pravděpodobnostní syntézu kombinující induktivní a deduktivní přístup. Následně jsem navrhl a implementoval rozšíření této metody o možnost použití rewardů. Má implementace vychází z projektu PAYNT (`https://github.com/gargantophob/synthesis`). Poté jsem se zaměřil na využití této metody pro pravděpodobnostní syntézu, kde je pro specifikaci hledaného řešení použita until vlastnost. Došel jsem k závěru, že daná metoda pro tento účel lze použít bez větších změn. Rozšíření umožňující tuto specifikaci hledaného řešení jsem následně implementoval do téhož nástroje. Z provedeného experimentálního vyhodnocení vyplývá, že i po těchto rozšířeních daná metoda svou efektivitou převyšuje standardní způsob pravděpodobnostní syntézy, a to až o několik řádů.

Do budoucna je možné se zaměřit na rozšíření možnosti specifikace hledaného řešení pomocí kompletní PCTL logiky. Vhodné rozšíření by například bylo umožnění specifikace pomocí bounded until vlastností. Také se nabízí rozšíření daného nástroje o možnost vytvářet více protipříkladů z jednoho řešení které nesplňuje specifikované požadavky.

Literatura

[1] Ábrahám, E., Becker, B., Dehnert, C., Jansen, N., Katoen, J.-P. et al. Counterexample generation for discrete-time Markov models: An introductory survey. In: *International School on Formal Methods for the Design of Computer, Communication and Software Systems*. 2014, s. 65–121.

[2] Andriushchenko, R. *Computer-Aided Synthesis of Probabilistic Models*. Brno, CZ, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: `https://www.fit.vut.cz/study/thesis/22997/`.

[3] Andriushchenko, R., Češka, M., Junges, S. a Katoen, J.-P. Inductive Synthesis for Probabilistic Programs Reaches New Horizons. 2021.

[4] Chrszon, P., Dubslaff, C., Klüppelholz, S. a Baier, C. ProFeat: feature-oriented engineering for family-based probabilistic model checking. *Formal Aspects of Computing*. 2017, sv. 30, č. 1.

[5] Dehnert, C., Junges, S., Katoen, J.-P. a Volk, M. The probabilistic model checker storm. *ArXiv preprint arXiv:1610.08713*. 2016.

[6] Forejt, V., Kwiatkowska, M., Norman, G. a Parker, D. Automated verification techniques for probabilistic systems. In: *International school on formal methods for the design of computer, communication and software systems*. 2011, s. 53–113.

[7] Gerasimou, S., Calinescu, R. a Tamburrelli, G. Synthesis of probabilistic models for quality-of-service software engineering. *Automated Software Engineering*. 2018, sv. 25, č. 4.

[8] Kwiatkowska, M., Norman, G. a Parker, D. Stochastic model checking. In: *International School on Formal Methods for the Design of Computer, Communication and Software Systems*. 2007, s. 220–270.

[9] Kwiatkowska, M., Norman, G. a Parker, D. PRISM 4.0: Verification of Probabilistic Real-Time Systems. 2011, s. 585–591.

[10] Kwiatkowska, M., Norman, G. a Parker, D. Probabilistic verification of Herman's self-stabilisation algorithm. *Formal Aspects of Computing*. 2012, č. 4.

[11] Kwiatkowska, M., Norman, G. a Segala, R. Automated Verification of a Randomized Distributed Consensus Protocol Using Cadence SMV and PRISM? In: Springer. *International Conference on Computer Aided Verification*. 2001, s. 194–206.

[12] Lakin, M. R., Parker, D., Cardelli, L., Kwiatkowska, M. a Phillips, A. Design and analysis of DNA strand displacement devices using probabilistic model checking. *Journal of the Royal Society Interface*. The Royal Society. 2012, sv. 9, č. 72, s. 1470–1485.

[13] Quatmann, T., Jansen, N., Dehnert, C., Wimmer, R., Ábrahám, E. et al. Counterexamples for expected rewards. In: *International Symposium on Formal Methods*. 2015, s. 435–452.

[14] Češka, M., Hensel, C., Junges, S. a Katoen, J.-P. Counterexample-Driven Synthesis for Probabilistic Program Sketches. *Lecture Notes in Computer Science Formal Methods – The Next 30 Years*. 2019.

[15] Češka, M., Jansen, N., Junges, S. a Katoen, J.-P. Shepherding Hordes of Markov Chains. *Tools and Algorithms for the Construction and Analysis of Systems Lecture Notes in Computer Science*. 2019.